THE PLANNING COORDINATOR: A DESIGN ARCHITECTURE FOR AUTONOMOUS ERROR RECOVERY AND ON-LINE PLANNING OF INTELLIGENT TASKS

by

Jeffrey J. Farah

Rensselaer Polytechnic Institute
Electrical, Computer, and Systems Engineering Department
Troy, New York 12180-3590

December 1992

CIRSSE REPORT #134

Table of Contents

Section Page

Section Page

List of Figures

1.0 Introduction

This chapter introduces the motivation for the proposed research. It establishes the problem domain, presents the problem statement and presents the organization of the remainder of the document.

1.1 Motivation

Conceptually, the goal of any automated system has two goals. The first goal is to be capable of performing specified, repetitive tasks more quickly and efficiently than a human predecessor or counterpart could. Here quickly is defined in terms of speed, and efficiently is defined in terms of quality, reliability, and cost. The second goal is to be capable of performing the first goal in environments not conducive to the health or growth of human beings.

The goal of an *x-autonomous* system, where *'x'* is a qualitative representation of the degree of system autonomy, is to perform the previously delineated tasks in a robust, competent manner with varying degrees of human intervention. This human intervention can range from complete human control, *teleoperation* or *telepresence*, through partial human intervention, *semi-autonomy* or *telesupervision*, to no human interference, *complete system autonomy* or simply *autonomy*, except when specifically requested by the automated system.

The goal of an intelligent *x-autonomous* system is not only to perform a previously delineated set of tasks. It must also be able to perform related and/or unrelated tasks. These tasks may be the result of a physical and/or logical change in the *x-autonomous* system, a change in the operation of the *x-autonomous* system, and/or a change in the environment in which the *x-autonomous* system operates.

The ability to adapt to environmental changes, specifically from a structured environment to an unstructured environment is of paramount importance in the development of intelligent *x-autonomous* systems. It must be able to generate new task level plans and competently recover

from task level errors generated during task level operations. Without this capability, the relating of the known to the unknown is not possible. Without the ability to relate the known to the unknown, intelligence is also impossible to obtain. Webster's Ninth New Collegiate Dictionary defines *intelligence* in two ways as below:

1. *the ability to learn or understand or deal with new or trying situations,*
2. *the ability to apply knowledge to manipulate one's environment.*

To manipulate one's environment requires an understanding of the representation of the environment. An environment can be considered to be a structured or unstructured relation of arbitrarily complex objects and events that themselves may or may not be related. These events (objects), although appearing to be a continuous stream of occurrences, can be viewed in all cases, as sequences of discrete incidences. Viewed as discrete occurrences regardless of complexity, complex discrete events can be decomposed into smaller, less complex, discrete events .

Because an environment is changeable, the events that comprise it may be considered dynamic. Hence the overall environment can be considered a *D*iscrete *E*vent *D*ynamic *S*ystem (*DEDS*). By definition, a *DEDS*, is a system that can be represented by an enumerable number of states. Graphically, a *DEDS*, can be most succinctly represented by a *G*eneralized *S*tochastic *C*olored *P*etri *N*et (*GSCPN*). Analytically, the *GSCPN*, is an extension of Ordinary Petri Nets and has a strong mathematical foundation. This foundation has permitted the development of numerous, reliable analysis tools that are used to ensure the validity of the *GSCPN* properties. However, a *GSCPN*, alone, lacks the relational information necessary to distinguish between similar (i.e., non-identical) events, identical events, or completely unrelated events. This inability requires a separate *GSCPN* representation for each non-identical event. This need for many physically distinct but logically equivalent representations creates the potential for state-space explosion. During the attempted representation of a non-trivial environment and during complicated task

execution within an environment, the introduction of new information may be required. In establishing the necessary new representations, state-space explosion may also occur. This limitation of the *GSCPN* is eliminated through the use of Semantic Networks with hierarchically distributed sub/superclass relations.

Like the *GSCPN*, the Semantic Network (*SNet*) contains nodes, representing conceptual units, and directed links, representing actions that relate one conceptual unit to another. The major difference between *GSCPN*s and *SNet*s is that *SNet*s, due to their structure of hierarchical abstraction and relational connectivity, can be used for inference as well as understanding, while *GSCPN*s cannot. The relational quality of the *SNet* provides a natural limiting agent that prevents the state-space explosion described previously for a *GSCPN*. Further, through the application of linguistic weighting values appearing as fuzzy weights rather than crisp number weights, a more natural and *human* relational hierarchy is possible.

The primary goal of this thesis is to develop a robust architecture that incorporates the strengths of existing representation and reasoning methodologies such as *GSCPN*s, *SNet*s, Fuzzy Logic, Expert Systems, and Object Orientation, and results in an intelligent, stand alone *x-autonomous* system.

1.2 Statement of Problem

Developing a robust, task level, error recovery and on-line planning architecture is an open research area. There is previously published work on both error recovery and on-line planning; however, none incorporates error recovery and on-line planning into one integrated platform.

The integration of these two functionalities requires an architecture that possesses the following characteristics. The architecture must provide for the inclusion of new information without the

destruction of existing information. The architecture must provide for the relating of pieces of information, old and new, to one another in a non-trivial rather than trivial manner (e.g., object one is related to object two under the following constraints, versus, yes, they are related; no, they are not related). The architecture must provide for the functional identification and restriction of data types (e.g., Object Type, cylinder; Restriction, circular top and bottom possessing length). Finally, the architecture must be not only a stand alone architecture, one that is capable of functioning on its own, but also one that can be easily integrated as a supplement to some existing architecture.

This thesis proposal addresses architectural development. Its intent is to integrate error recovery and on-line planning onto a single, integrated, multi-processor platform. This *intelligent x-autonomous platform*, called the Planning Coordinator, will be used initially to supplement existing *x-autonomous* systems and eventually replace them.

1.3 Document Organization

The remainder of this thesis proposal is organized into the following sections and subsections:

Section 2: Background Theory and Definitions

This chapter provides background theory and definitions that are relevant to the constructs utilized in later chapters of the thesis. This chapter is broken up into six major subsections:

2.1 Autonomous Intelligent Systems: This subsection examines intelligent system developments and their characteristics.

2.2 Error Recovery in Automated Systems: This subsection examines previous error recovery schemes used in automated systems. Included in this discussion are operating system error recovery techniques and Petri Net error recovery schemes.

2.3 Fuzzy Logic: This subsection examines the use of Fuzzy Logic estimators to provide a qualitative versus quantitative situation evaluation mechanism. In particular, Fuzzy Rule Base Generation, and Fuzzy Cognitive Map Summation Techniques are reviewed.

2.4 Generalized Stochastic Colored Petri Nets: This subsection provides background material on Generalized Stochastic Colored Petri Nets and develops the use of such petri nets as a means of graphically representing the execution sequence of Discrete Event Dynamic System tasks, specifically error recoveries and on-line plan generations.

2.5 Semantic Networks: This subsection provides background material on the Semantic Network construct. In particular, previous work on both the limitation of state-space explosion utilizing the Semantic Network construct and the utilization of search techniques for searching a Semantic Network are examined. Finally the hierarchical abstraction of conceptual units is examined with particular focus on the linguistic weighting of inter-node links within a Semantic Network.

Section 3: The Planning Coordinator

This section introduces the Planning Coordinator and defines its Macro Architecture, Internal and External Interfaces, and Functional Description in three major subsections:

3.1 Intelligent Machine Model: This subsection outlines the Intelligent Machine Model and describes the flexibility which has made it the choice for the initial application of the Planning Coordinator.

3.2 Planning Coordinator Macro Architecture: This subsection details the macro-architecture of the Planning Coordinator and outlines its component structure, role, physical location and logical location within the hierarchy of an intelligent *x-autonomous* system.

3.3 Planning Coordinator Communication and Interface Description: This subsection details the internal logical interfaces between components of the Planning Coordinator as well as the logical interfaces necessary to integrate the Planning Coordinator into existing intelligent *x-autonomous* systems.

3.4 Functional Description: This subsection presents the functional description of the Planning Coordinator, detailing the operation of the Planning Coordinator.

3.5 Example Operation of the Planning Coordinator: This subsection provides a comprehensive example of the operation of the Planning Coordinator. The environment in which the Planning Coordinator functions is limited to the Center for

Intelligent Robotic Systems for Space Exploration testbed for this example and will be explained.

Section 4: Research Goals and Proposed Work

This section describes the research goals and proposed work for the completion of this Thesis. Included in this chapter are contributions to date as examined through the examples of *Section 3.5*, and expected contributions.

References and Bibliography

Provides a listing of the literature cited as well as the literature reviewed during the researching of this topic. It includes work to date by this student as well as previous work by other researchers.

2.0 Background Theory and Definitions

This section introduces the background theory and definitions that are relevant to the constructs utilized in later chapters of the thesis. The background theory introduces previous work in the subject areas and examines the differences between the approaches being taken by other researchers and the proposed approach.

2.1 Autonomous Intelligent Systems

Autonomy, as defined in Webster's Ninth New Collegiate Dictionary, is *the quality or condition of functioning independently of other parts*. Autonomy is not limited to a single biological or mechanical entity.

Intelligence, as defined in Webster's Ninth New Collegiate Dictionary, is *the ability to learn or understand or deal with new or trying situations, or the ability to apply knowledge to one's environment.* Intelligence is not limited by definition to the human species or to biological species. To date, however, the reality has been that biological entities are the only entities that have exhibited intelligence to *any* degree. A significant amount of work has gone into research specifically designed to create an intelligent non-biological entity.

A *System*, as defined in Webster's Ninth New Collegiate Dictionary, is *a set of facts, principles, rules, etc. classified or arranged in a regular or orderly form so as to show a logical plan linking the various parts.* A system is not limited to a mechanical or biological device capable of motion, where motion is the movement of any part. Much work has been dedicated to the creation of various systems that address various problems.

There is neither a Webster's New Ninth Collegiate Dictionary definition for an Autonomous Intelligent System, nor a coherent literature definition that adequately incorporates the separate

definitions of autonomy, intelligence and system. To provide a coherent definition for the purposes of this thesis proposal, the following is offered:

Autonomous Intelligent System:

A collection of objects, facts, or principles arranged in an ordered physical and/or logical form that:

1. Can function independently of other systems but can also be configured to perform cooperative action with other systems.
2. Deals with new or trying situations and learns from its attempts to apply its knowledge to its environment.
3. May or may not possess the ability of locomotion (i.e. the movement of any of its parts or the movement of its entire structure) and may or may not be biological.

This definition permits the inclusion of intelligent autonomous robotic systems as well as intelligent autonomous non-robotic systems under the umbrella of intelligent autonomous systems. Note that an autonomous system can, if necessary, request assistance when needed. This does not invalidate its autonomous nature. The consolidation of locomotive and non-locomotive intelligent systems under the same umbrella is a necessary undertaking as the two areas often posses specific techniques that are applicable to both, but are overlooked merely because they are not part of the same general category.

There has been considerable work in the attempted identification and development of intelligent autonomous systems. The following subsection introduces two such attempts and examines their strengths and weaknesses.

2.1.1 Intelligent Machine Architectures

Research into the development of Intelligent Robotic and Machine Architectures (IRMA) has led to the conclusion that hierarchically distributed system architectures more adequately represent intelligence than do other models [Albus 75]. As discussed by Saridis [Saridis 79] and further confirmed by Albus [Albus 83], intelligent robotic systems obey the principle of Increasing Precision with Decreasing Intelligence (IPDI). This principle indicates that as intelligence increases, the need for precise detail decreases in favor of conceptual combinations of more abstract ideas. Similarly, as intelligence decreases, the need for more precise detail is needed as opposed to conceptualizations. However, not all intelligent autonomous systems are necessarily robotic systems as a robot is defined to be, *a machine that can sense and react to input and cause changes in its surroundings with some degree of intelligence and ideally without human intervention.* An intelligent autonomous system may be any system which conforms to the definition of Section 2.1. Nevertheless, it still obeys the IPDI principle.

Albus [Albus 90a, Albus 90b] proposes a hierarchy for an intelligent system in which there are seven basic elements: *actuators, sensors, sensory processing, world modeling, task decomposition, value judgment, and global memory/communications.* This proposed hierarchy is based extensively on Saridis' theory of intelligent controls [Saridis 85] and Brooks' layered control system for a mobile robot, [Brooks 86]. Albus postulated that :

- Control bandwith decreases about an order of magnitude with each higher level.
- Perceptual resolution of spatial and temporal patterns contracts about an order of magnitude at each higher level.
- Goals expand in scope, and planning horizons expand in space and time about an order of magnitude at each higher level .
- Models of the world and memories of events expand in space and time by about an order of magnitude at each higher level.
- At each level, tightly coupled functional modules perform task decomposition, world modeling, sensory processing, and

value judgment, with feedback control loops closed at every level.

The first three postulates are consistent with the development of the Intelligent Machine Model by Saridis and coworkers [Saridis 77], [Saridis 83]. However, the last two postulates introduce potential difficulties in the implementation of the Albus architecture. Albus proposes maintaining world models and memories at each level of his hierarchy. While this would be ideal, there is the very likely problem that world model coherence problems will arise, as the update of *seven* world models, one at each of the seven levels of his architecture, is necessary. In addition, limited memory availability in an intelligent autonomous system is a constraint that cannot be ignored. Redundant storage of information is not acceptable under such conditions. Albus also proposes that value judgments (undefined), sensory processing and task decomposition be performed at each level, with closed feedback control loop execution. While sensory processing is arguably necessary at each level, task decomposition and value judgment are not. This is due to the IPDI principle. By the time a task is sent to a lower level, the task has already been decomposed from the perspective of an overall task description. If the lower levels execute the commands requested of them, this results in a time and execution space saving (i.e., RAM necessary for holding the executing program).

The Intelligent Machine Model (IMM) developed by Saridis and coworkers divides the functions of an intelligent machine into three major sections wholly based on the IPDI principle. Divided into the Organization Level, the Coordination Level, and the Execution Level, Saridis attempts to functionally consolidate the intelligent machine's operations. The three levels are examined below,

Organization Level:

Performs planning and high level decision making from long term memories. Requires large quantities of knowledge processing but little or no precision.

Coordination Level:

Serves as an interface between the Organization Level and the Execution Level. Involves coordination, decision making, and learning on a short term memory.

Execution Level:

Executes the control functions passed to it from the Coordination Level on the hardware comprising the Intelligent Machine.

The generality of the Intelligent Machine Model permits the introduction of a more consolidated description of the different modules that comprise an Intelligent Machine. In addition, it facilitates the inclusion of new modules into the overall Intelligent Machine architecture.

Neither the Albus model nor the IMM model provides for intelligent error recovery or on-line planning directly. However, the ease with which a module that performs these tasks can be included is of great importance. Hence choosing an intelligent system architecture into which the Planning Coordinator is to be incorporated, becomes a choice of the architecture which most easily facilitates the introduction and inclusion of new modules. For this reason the Intelligent Machine Model has been chosen.

The following section introduces previous work in intelligent error recovery and outlines identified strengths and weaknesses.

2.2 Intelligent Error Recovery in Automated Systems

Intelligent error recovery is akin to the generation of an alternate plan given that an unexpected though potentially predictable event has occurred. If it were possible to enumerate all of the potential errors, store a recovery plan for each error, and choose the optimal error recovery plan efficiently each time an error occurs, intelligent error recovery would be simple. This, however, is not the case. It is not feasible to employ a brute force method which systematically generates and stores all of the exhaustively enumerated error recovery paths for a series of tasks. For even

a mildly complex task, the number of possible errors and hence differing error recoveries is inordinately large, requiring a tremendous amount of storage. Because of this lack of feasibility, intelligent error recovery is a necessity. Presently, intelligent error recovery methodologies for automated systems provide a wide range of potential solutions, each with its own limitation(s). Presented below are several representations for error recovery that have gained notoriety. In each case, the strengths and/or weaknesses of the methodology are examined.

2.2.1 AND/OR Graph Representation

Any autonomous system plan can be expanded to include all of the potential errors that could result. This exhaustive enumeration can be represented as an AND/OR graph that in artificial intelligence [Nilsson 80] has been used to structure the goals and subgoals, preconditions, of a process plan. Regarding error recovery, AND/OR graphs have been used by de Mello and Sanderson [de Mello 86], [de Mello 88], [Sanderson 87a], [Sanderson 87b], and Cao and Sanderson [Cao 91].

The algorithm capable of developing an AND/OR graph representation ensures that a solution will be found *if one does exist* since all possible plans are enumerated. However, as the complexity of an environment grows, and the complexity of a task grows, the AND/OR graph also grows, exponentially. The exponential growth is in the number of possible nodes and branches that must be explored to obtain a solution. As a result, the state-space that the AND/OR graph represents suffers from state-space explosion. The searching of the AND/OR graph for a potential solution will likewise increase as will the required storage space for the AND/OR tree generated from the AND/OR graph. Hence when an error recovery plan is needed, the time to search the given state-space for a solution may be inordinately large. In addition, if the AND/OR graph is itself large and the storage space of the autonomous system in which the AND/OR graph resides is too small to accommodate the entire graph, pruning may be necessary.

Unfortunately, as the AND/OR graph is pruned, it is possible that *the* potentially successful error recovery plan will be pruned as well.

2.2.2 Failure Reason Analysis

Failure reason analysis as described by Srinivas [Srinivas 76], [Srinivas 78] attempts to find an explanation(s) for a failure(s). As the explanation is determined, the reasons for the failure are classified into four separate categories. Using knowledge of the preconditions of an activity and its expected post conditions, Srinivas' system examines these categories: *operation errors*, *information errors*, *precondition errors*, and *constraint errors*. It then implements a backtracking scheme to recover from the failures. The backtracking produces a failure tree which is analyzed to determine the explanation(s) for the cause(s) of the failure. Only after the analysis of the failure tree is an error recovery strategy generated and attempted.

Like the AND/OR graph representation, there is the potential for combinatorial state space explosion. This is because during the backtracking procedure, each precondition of the activity in the failure tree must be expanded to find the probable cause of the failure. Unlike the AND/OR graph representation, this methodology is limited to the generation of one error recovery scheme at a time. Nevertheless, the time needed to generate the failure tree, perform the analysis on the tree and generate the error recovery scheme is prohibitive. In addition, this representation, like the AND/OR graph representation, does not provide for the inclusion of new information into the operating environment. Rather, it relies on the existence of a static environment.

Chang [Chang, 89a, Chang 89b] improved on Srinivas' approach by taking advantage of the hierarchical structure of the plan activities in the generation of the failure tree. While this

proved to provide a substantial reduction in the effort required to search the failure tree, it did not reduce the difficulty of state-space explosion.

2.2.3 Sensor Control Software Methodology

Lee and coworkers, developed a sensor based approach that would diagnose task errors through the examination of a sensor signature. The sensor signature is defined as a collection of relevant sensor values *expected to occur* at a given point during a robot task cycle [Lee 83a] [Lee 84] [Lee 85]. This approach, while a potentially successful one, does not account for the potential of misreadings, miscalibrations, or internal data corruption of the autonomous system. In addition, it requires the mounting and identification of a significant number of additional sensors that must be maintained. Finally it expects well-defined *à priori* knowledge of the sensors such as location, orientation, and potential positions.

2.2.4 Rapid System Reconfigurability

Williams regards error recovery from the perspective of rapid system reconfigurability [Williams 86]. Using a state table to represent sensor driven control, Williams utilizes a control program to match existing states to those in the state table. This technique requires an operator to teach the system how to react in a given situation. Once the system is taught, it can recover from errors. There are two major difficulties with the Williams approach. First, in a real workcell, the reconfiguration of sensors may not be possible. Second, complex sensory systems such as vision systems have been shown by Lee [Lee 87] to be difficult to integrate into sensor driven control.

2.2.5 Real Time Monitoring of Robot Control

As delineated by Gini and Smith [Gini 83], [Smith 86], [Gini 85a], [Gini 85b], and [Gini 85c], a knowledge-based monitoring system that watches the preconditions and postconditions of each robotic instruction can reduce the burden of rigorous off-line programming of robots designed to provide robust error recovery. Unlike their contemporaries, Williams and coworkers, Gini and Smith researched an approach that would have the ability to keep a robotic task history and use the history to perform diagnosis and error recovery. The approach utilized three procedures: a *Preprocessor*, an *Augmented Program Processor*, and a *Recoverer*.

Were an error to occur, the Gini and Smith system uses the history and information available about the task for diagnosis. Once the diagnosis is completed, the system appends additional procedures to the existing augmented program to perform an error recovery to the next valid state. Of the systems examined so far, this system is the only one that can interpret the error and specify relationships among varying objects within the operational environment. The major limitation of this model is that it will only use one strategy to recover from an error. That strategy requires historical knowledge for diagnosis of the error and generation of the strategy. This implies an unchanging environment or one in which historical knowledge is sufficient to enact error recovery. For a completely identified, structured environment this approach may be sufficient. For a dynamically changing one, it is not.

2.2.6 Plan Feasibility Model

Brooks [Brooks 82] has developed a *plan feasibility model* to determine the feasibility of some intelligent plan. This feasibility model determines whether a plan is feasible or not, given knowledge of the intelligent system and environment. The approach relies on a geometric database to infer the effects of actions and the propagation of errors. This model is more of a

pre-processor of the plan rather than an error recovery mechanism. However with a sophisticated sensing capability, the model may become useful for error recovery. Unfortunately, the sophisticated sensing combined with the programming to interpret the sensory data may make the model prohibitively costly, due to the cost of sophisticated sensing devices applicable to the task, and slow due to the interpretation of the data obtained from the sensing equipment.

2.2.7 Causal Reasoning Model

The Causal Reasoning Model is an error recovery strategy developed by Kumaradjaja and DiCesare [Kumaradjaja 89]. This model is very similar to the AND/OR graph representation of de Mello and Sanderson. Both models incorporate all possible paths from the initial state to the desired state. The Causal Reasoning Model differs from the AND/OR graph representation in that it incorporates the hierarchical structures of the activities and states. This structuring permits some guidance as to the appropriate sequence of error recovery activities.

Additionally, like the AND/OR graph representation, new information is not easily handled by the Causal Reasoning Model. Further, uncertainties that exist in the environment or in the robotic model are unaccounted for. Finally, it is expected by the Causal Reasoning Model that a structured, well-known environment exists since the Causal Reasoning Model relies on an external entity to provide it with structured knowledge. Such an entity is not always available. In addition, uncertainty is always present in an environment. It is here, specifically, that the Causal Reasoning Model is insufficient.

2.2.8 Operating System Techniques

By definition, an operating system provides the environment within which programs are executed [Silberschatz 88]. As such there are several types of services that an operating system will necessarily provide. Among those services are error detection and error correction.

Under error detection the Operating System must be aware of errors in :

Hardware
- CPU
- Memory - RAM and ROM
- Controllers - HDU, FDU, ODU, DMA, etc.
- I/O Devices - Printer, Tape Unit , etc.

Software
- Scheduling
- I/O Drivers
- User Programs
- Security Programs
- etc.

In general, error recovery by an operating system is a more complex task. There has been considerable research done in the attempt to create a fully robust operating system capable of determining and recovering from *task level* errors autonomously. To date, the only operating systems that perform *task level,* autonomous error recovery are those that function to govern the operation of a distributed processor system. The recovery methodology is similar to that employed by Williams' *Rapid System Reconfigurability* technique. It is limited in scope to the determination of a processor or link failure through a handshaking time-out procedure. Once the identification of a failure has been verified, a rerouting is accomplished to bypass the failed entity. The operating system does not make an attempt to fix the failed entity. This is suitable for handling partially redundant tasks, where two devices (i.e., sensors) of similar functionality

each handle a portion of the load until one fails. On the failure, the other device takes the full load.

Single processor operating systems, at present, detect errors or errant situations. If the errors are communication errors (i.e., can be fixed through error correcting codes) the operating system will autonomously fix them. If the errors are due to user software (i.e., protection violation, security violation, division by zero, etc.) the operating system catches them and fixes them through the use of interrupt or trap service routines. These routines compare the *violation* against those in a look up table, in a manner similar to that employed by Williams in his state table look up.

Specific operating systems such as UNIX™ employ certain techniques. For example rather than dealing with a 'pathological condition' through the use of elaborate control algorithms, UNIX™ will perform a controlled crash of the system, called a *panic* [Ritchie and Thompson 74], [Thompson 78]. As per Thompson [Thompson 92], "There is no real error recovery strategy in UNIX™.....Usually one has to design on a line between two extremes of speed and reliability. Speed usually involved a lot of caching and asynchronous I/O. Reliability usually involved synchronous guaranteed recoverable I/O. We carefully thought about these problems and almost always came down on the side of speed. For our uses....faster access with occasional crashes was the usual choice..."

With respect to file identification and directory identification (devices are considered special files), UNIX™ uses *file structures* and *inode structures*. A file descriptor is used by the UNIX ™ kernel as an index into a *file structure*. The file structure points to an *inode structure*. This inode structure is allocated out of a fixed-length table. This table is an 'in-core' copy of the master table which is located on disk. Discrepancies in this table are handled through comparison to the master copy on disk. Discrepancies between the inode structure table and the

file structure table are determined through the comparison of fields that contain reference counts of the number of file structures / inode structures pointed to /from.

2.2.9 Summary

Each of the intelligent error recovery methodologies presented in this section have been shown to be applicable under limiting conditions whether those conditions be due to space limitations, time limitations, search inefficiencies, or environmental expectations. In contrast to the techniques presented so far, the goal of the Planning Coordinator is to demonstrate the ability to employ strengths equivalent to the aggregate of the strengths of the techniques presented so far, without a corresponding aggregate of their weaknesses. The Planning Coordinator is discussed in Section 3.

2.3 Fuzzy Logic

Fuzzy logic was created by Lotfi A. Zadeh in the mid-1960's in response to a need for bridging the gap between the way human beings reason and the way computers are programmed. Expert Systems were introduced in the early 1960's as a means of assisting human operators by performing the function of an 'expert' in a given field. The following subsections introduce Fuzzy Logic terms and theory for later use in this thesis proposal.

2.3.1 Fuzzy Logic Definitions

Concepts defined by Zadeh, [Zadeh 84];

> Fuzzy Logic
> A kind of logic using graded or qualified statements rather than ones that are strictly true or false. The results of fuzzy reasoning are not as definite as those derived by strict conventional logic, but they cover a larger field of discourse.

Fuzzy Set

A set in which the transition from membership to non-membership may be gradual rather than sharp. The degree of membership is specified by a number between one (full membership) and zero (non-membership).

Fuzzy Modifier

Operation that changes the membership function of a fuzzy set by spreading out the transition between full membership and non-membership, by sharpening the transition or by moving the position of the transition region.

Linguistic Variable

Ordinary-language terms that are used to represent a particular fuzzy set in a given problem; such terms would include: *large*, *small*, *medium*, etc.

Ultrafuzzy Set:

A set whose membership function is itself a fuzzy set, so that an object in the set is assigned a range of grades rather than being given a membership grade between zero and one. For example a membership grade of *0.55* might become *0.4-0.6*.

A more formal definition of a fuzzy set and the operations applicable to a fuzzy set are given below. This formalization is taken from [Zadeh 84a]. Let Ξ be a space of points (objects), with a generic element of Ξ denoted by ξ, thus $\Xi = \{\xi\}$.

Fuzzy Set

A fuzzy set, Φ, is characterized by a membership or characteristic function $f_{\Phi}(\xi)$ which associates with each point ξ in Ξ a real number in the interval $\{0, 1\}$. The nearer the value of $f_{\Phi}(\xi)$ to one, the higher the membership grade of ξ in Ξ.

Empty Fuzzy Set

A fuzzy set is empty if and only if its membership function is identically zero on Ξ.

Equality

Two fuzzy sets Φ and Θ are equal, written $\Phi = \Theta$, if and only if $f_\Phi(\xi) = f_\Theta(\xi)$ for all ξ in Ξ.

Complement

The complement of a fuzzy set, Φ, is denoted by Φ' and is defined as: $f_{\Phi'} = 1 - f_\Phi$.

Containment:

The fuzzy set Φ is contained in the fuzzy set Θ if and only if $f_\Phi(\xi) \leq f_\Theta(\xi)$.

Equivalently, $\Phi \subset \Theta \Leftrightarrow f_\Phi(\xi) \leq f_\Theta(\xi)$.

It must be noted that although the membership function of a fuzzy set has some resemblance to a probability function when Ξ is a countable set, or a probability density function when Ξ is a continuum, a fuzzy set is completely non-statistical in nature. This point is clarified in the following discussion of the combining rules of fuzzy sets.

Union

The union of two fuzzy sets, Φ and Θ, with respective membership functions $f_\Phi(\xi)$, and $f_\Theta(\xi)$, is a fuzzy set X, written as $X = \Phi \cup \Theta$ whose membership function is related to those of Φ and Θ by the relation: $f_X(\xi) = Max[f_\Phi(\xi), \Theta(\xi)]$, $\xi \in \Xi$. Note that associativity holds here.

Intersection

The intersection of two fuzzy sets, Φ and Θ, with respective membership functions $f_\Phi(\xi)$, and $f_\Theta(\xi)$, is a fuzzy set X, written as $X = \Phi \cap \Theta$ whose membership function is related to those of Φ and Θ by the relation: $f_X(\xi) = Min[f_\Phi(\xi), \Theta(\xi)]$, $\xi \in \Xi$.

Note: Henceforward $f_\Phi(\xi)$ will be denoted f_Φ.

Algebraic Product

The algebraic product of Φ and Θ is denoted by $\Phi\Theta$ and defined by the relation: $f_{\Phi\Theta} = f_{\Phi} f_{\Theta}$, where $\Phi\Theta \subset \Phi \cap \Theta$.

Algebraic Sum

The algebraic sum of Φ and Θ is denoted $\Phi + \Theta$, and is defined by the relation $f_{\Phi+\Theta} = f_{\Phi} + f_{\Theta}$, where $f_{\Phi} + f_{\Theta} \leq 1$ for all $\xi \in \Xi$.

Absolute Difference

The absolute difference of Φ and Θ is denoted $| \Phi - \Theta |$ and is defined by the relation $f_{|\Phi-\Theta|} = |f_{\Phi} - f_{\Theta}|$.

Convex Combination

The convex combination of three arbitrary fuzzy sets, Φ, Θ and Π is denoted by $(\Phi, \Theta ; \Pi)$ and is defined by the relation $(\Phi, \Theta ; \Pi) = \Pi\Phi + \Pi' \Theta$, where Π' is the complement of Π. Equivalently $f_{(\Phi, \Theta ; \Pi)} = f_{\Pi} f_{\Phi} + [1 - f_{\Pi}] f_{\Phi}$ for all $\xi \in \Xi$.

Fuzzy Relation

An *n-ary* fuzzy relation is a fuzzy set in the product space of $\Xi \times \Xi \times \Xi \times \Xi \times \ldots \times \Xi \times \Xi$. For such relations, the membership is of the form $f_{\Phi}(\xi_1, \xi_2, \ldots, \xi_n)$ where $\xi_i \in \Xi$, $i = 1, \ldots, n$.

The following subsection outlines some of the theory of Fuzzy Logic and previous work utilizing Fuzzy Logic for error recovery.

2.3.2 Previous Work Using Fuzzy Logic for Error Recovery

Fuzzy Logic attempts to replace familiar bi-valent logic [and probability theory] with a continuum. A fuzzy set is a class with fuzzy boundaries. Such a class, as the definition in the previous section indicates, is characterized by associating a grade of membership in the class with every object that could be in the class. Human beings perform this assignment

subconsciously and without a full understanding as to the mechanisms whereby the assigned grades of membership are generated. The majority of computers and computer programs, to date, do not take advantage of the inherently fuzzy nature of the environment in which they operate. As a result, a computer has difficulty in recognizing the equivalence of two different individuals' handwriting, whereas a human being would not.

Previous work in Fuzzy Logic specifically in the area of error recovery and/or on-line planning is not extensive. Fuzzy Logic has been used extensively in the creation of controllers that more closely mimic a human operator. The first such application was a development by F.L. Smidth & Co. of Copenhagen for the control of cement kilns. Their results indicate that the use of the fuzzy controller provides a cost and fuel savings as well as a more stable product.

A second application of Fuzzy Logic to controls was implemented by Shoji Miyamoto and Seji Yasunoby of the Systems Development Laboratory of Hitachi, Ltd. in 1984 for the automatic operation of a train by means of a *predictive fuzzy logic controller*. Unlike other fuzzy logic controllers, a *predictive fuzzy logic controller* predicts the results of the execution of the actions given in the rules that govern the operation of the fuzzy logic controller. Since then, Japanese industry has applied fuzzy logic to a variety of product developments but still has not applied fuzzy logic to the areas of error recovery and/or on-line planning. The reason for this is that with the fuzzy logic applications attempted, error recovery per se has not been needed.

In limited applications, scientists and engineers at the Institute of Industrial Cybernetics and Robotics in Sofia, Bulgaria have applied fuzzy logic to robot vision. Using a fuzzy classifier to determine the degree of 'edginess' of the fuzzy edge of the seam that the welder is forced to follow, it is found that the center of the seam is determined more readily. This seam center is determined to be quite accurate, and often different from the center that would have been

determined by taking the edge of the gray scale threshold. Their project shows that the robotic arc welding system is more effective and permits more coherent welding.

Regarding error recovery and/or on-line planning, there is no literature that examines a direct attempt to incorporate fuzzy logic. The work that most closely resembles the use of fuzzy logic for the development of error recovery schemes is that by Cao and Sanderson [Cao 91] and [Cao 92]. Utilizing the Fuzzy Petri Net introduced in [Looney 88] and [Chen 90], Cao and Sanderson create a Fuzzy Petri Net from an AND/OR graph.

The Fuzzy Petri Net generated from the AND/OR graph, as claimed by Cao [Cao 91] does not exhibit the same state space explosion that an Ordinary Petri Net generated from an AND/OR graph could. This is due to numerical constraints and ordering precedence relationships. However, the procedure for generating the Fuzzy Petri Net requires the generation of an Ordinary Petri Net first to examine the liveness, safeness and boundedness of the overall plan representation. Hence even before the Fuzzy Petri Net is created, pruning of the AND/OR graph may eliminate viable error recovery options. Secondly, it is stated in [Cao 92] that *if the Petri Net is designed correctly, an altered firing sequence will eventually lead to a goal state by correctly executing an error recovery path.* There is no accounting for an incorrectly designed Petri Net. Further there is no provision for the potential of cascading errors and the mechanisms whereby a cascading error would be handled. This is of concern in an autonomous system where a cascading error could be catastrophic to the overall operation of the intelligent autonomous system. Finally, [Cao 92] the fuzzy markings of the Fuzzy Petri Net represent the degree of completion of the process to which the fuzzy marking pertains. What happens to the fuzzy marking during an error recovery is not expanded upon.

2.3.3 Summary

The use of Fuzzy Logic has been developed to provide a more qualitative approach to affecting the operation and control of a system. In general, Fuzzy Logic has been used as a viable alternative to standard control system methods as evidenced by its use in the development of products in industry. As yet, there is little use of Fuzzy Logic for the development of robust error recovery and on-line planning schemes. The work that has been done exhibits problems not necessarily related to the use of Fuzzy Logic but which affect the overall system. No work has been uncovered that uses Fuzzy Logic in a manner similar to the work being proposed for this Thesis. The following section examines Generalized Stochastic Colored Petri Nets, their background and previous work using Petri Nets for error recovery.

2.4 Generalized Stochastic Colored Petri Nets

A Petri Net may be thought of as a graph theoretic abstract modeling concept used to efficiently model the states, preconditions, and functions of a discrete event or a continuous event dynamic system, particularly when concurrence and conflict are involved. Petri Net (PN) theory utilizes a highly graphical representation to exhibit a strong mathematically founded system design and analysis methodology. The strong mathematical foundation of PNs has permitted the establishing of reliable performance and analysis tools. These tools, combined with the event driven nature of the Petri Net construct, permit the use of Petri Nets as an effective means of modeling any discrete event dynamic system, or any continuous dynamic system in which the continuous system can be modeled as a continuum of discrete events. Overviews of the basic Petri Net Construct are provided in [Murata 84], [Murata 89] and [Peterson 81]. The following sections introduce the basic definitions and terms used in Petri Net theory and expands the terms to include stochastic and colored Petri Net extensions .

2.4.1 Petri Net Definitions

2.4.1.1 Ordinary Petri Net (OPN)

An Ordinary Petri Net , Λ, is a quadruple: $\Lambda = (P, T, F, M^0)$, where

- $P = \{P_1, P_2, P_3, \ldots P_p\}$ an enumerable set of places.
- $T = \{T_1, T_2, T_3, \ldots T_t\}$ an enumerable set of transitions.
- $F \subseteq I \cup O$, the flow relation where,
  I: $P \times T \rightarrow \{0,1\}$ describes the directed connectivity from places to transitions.
  O: $T \times P \rightarrow \{0,1\}$ describes the directed connectivity from transitions to places.
- M^0: describes the initial marking, distribution of tokens, within the Petri Net.
- $P \cap T = \varnothing$ and $P \cup T \neq \varnothing$

2.4.1.2 Arc Multiplicity

An OPN permits simple connectivity between places and transitions. An extension to the OPN permits non-simple arc weighting. When non-simple, arc weights are permitted the resulting models are called Generalized Petri Nets (GPN) or more simply Petri Nets. The definition of the OPN is generalized to include the non-simple arc weighting as below:

From:

- $F \subseteq I \cup O$, the flow relation where,
  I: $P \times T \rightarrow \{0,1\}$ describes the directed connectivity from places to transitions.
  If $I(P, T) = \kappa$, and $\kappa \neq 0$ then the arc weight from input place P to transition T is of weight κ.

  O: $T \times P \rightarrow \{0,1\}$ describes the directed connectivity from transitions to places.

If $O(T, P) = \kappa$, and $\kappa \neq 0$ then the arc weight from transition T to output place P is of weight κ

2.4.1.3 Place and Transition Ordering

The definition of the OPN as given implicitly describes an ordering of places and transitions since each place and transition is considered to be an independent entity. It is natural to order the places and transitions numerically. This convention will be followed throughout the remainder of this proposal.

Ordered numerically, a sequence of places or transitions can be considered more succinctly in vector notation with $\underline{\mu}^0$ replacing M^0 in the definition. This representation introduces the use of matrix notation to more easily represent the Petri Net.

2.4.1.4 Incidence Matrix

Using the definition of a GPN a matrix notation can be defined that more readily exhibits the connectivity described in the flow relation. This matrix description is given below.

Incidence Matrix

Given a GPN with a finite number of places, *p*, and a finite number of transitions, *t*, the *p x t* incidence matrix $\underline{C}$ is formed as follows:

$\underline{C}^-$ is the matrix of input arc weights, where $C_{i,j}^-$ is the number of input arcs from place P_i to transition T_j: $C_{i,j}^- = I(P_i, T_j)$.

$\underline{C}^+$ is the matrix of output arc weights, where $C_{i,j}^+$ is the number of output arcs from transition T_j to place P_i: $C_{i,j}^+ = O(T_j, P_i)$.

$\underline{C}$ is the incidence matrix given by $\underline{C} = \underline{C}^+ - \underline{C}^-$

2.4.1.5 Firing Rules and Firing Sequences

The flow of tokens through a GPN is regulated by the firing rules associated with the transitions. Given a GPN with some current marking $\underline{\mu}$, a transition T_j is said to be enabled if $\mu_i \geq I(P_i, T_j)$, $\forall_i = 1,.....,p$.

Transition Enabled Set

The transition enabled set of a GPN having some marking $\underline{\mu}$ is

$$ES(\Lambda, \underline{\mu}) \equiv \{t \in T \mid \mu_i \geq I(P_i, T);\ \forall_i = 1,2,....,p\}$$

From the current marking any transition in the enabled set may fire, whereby tokens are first removed from the transition's input place(s) and then are deposited in the transition's output place(s). Firing the transition T_j results in the new marking μ_i' given by.

$$\mu_i' = \mu_i - I(P_i,T_j) + O(T_j,P_i);\ i=1,2,.....,p$$

Firing a single transition is generalized to firing multiple transitions in sequence as follows. Let σ represent an ordered sequence of transition firings. Let $\sigma|_T$. represent the number of occurrences of the transition T in σ. The firing count vector, denoting the sequence of firing is obtained as follows.

$$\underline{\sigma} \equiv [\sigma|_{T_1}\ \sigma|_{T_2} ....\sigma|_{T_t}]^T$$

Therefore from the current marking, $\underline{\mu}$, firing the transitions sequence represented by $\underline{\sigma}$ results in the new marking $\underline{\mu}'$ as below:

$$\underline{\mu}' = \underline{\mu} + \underline{C} \cdot \underline{\sigma}$$

2.4.1.6 Reachability Graph

The states of a GPN are associated with the results of applying the firing rules and firing sequences previously described to the initial marking of the GPN. From any GPN a graph called the reachability graph can be generated. This graph is composed of a set of states and a set of directed links that indicate legitimate state changes through specific transition firings. As the name implies, a reachability graph for a particular state, indicates the subsequent states that can be reached given a particular transition firing sequence. The reachable states compose a set known as the reachability set.

It is possible that for some GPNs, the reachability graph will be unbounded (i.e., the cardinality of the reachability set is infinite). Since the problem of generating error recovery or on-line plans assumes an enumerable number of states, it is assumed that the reachability set is finite. If necessary, tests for the boundedness of a GPN can be performed. Formally these tests are as given below.

- The bounded reachability set for a GPN, Λ, is given as.
  $RS(\Lambda,\mu^0) \subset N_{0+}{}^p$
- $RS(\Lambda,\underline{\mu}^0) \Leftrightarrow \exists \underline{\sigma} \in \Sigma(\Lambda,\underline{\mu}^0) \ni \underline{\mu} = \underline{\mu}^0 + \underline{C} \cdot \underline{\sigma}$
- $|RS(\Lambda,\underline{\mu}^0)| < \infty$

Note that structural boundedness means that a GPN is bounded for any finite initial marking.

2.4.1.7 Generalized Stochastic Petri Nets

Generalized Stochastic Petri Nets (GSPN) are a time domain extension of GPNs. Each transition of a GSPN is associated with an exponential density function that describes the probability of the time required to fire the transition. This is accomplished by augmenting the definition of a GPN with a parameter describing the exponential distribution for each transition.

Hence a GSPN, Λ, is a quintuplet where $\Lambda = (P, T, F, \Gamma, M^0)$, and $\Gamma: T \rightarrow R_+ \cup \infty$

The exponential family of density functions has infinite support. Hence the reachability graph for a GSPN is formed by considering not the GSPN but the underlying GPN. The reachability set contains all of the states that are possible, and also contains the directed links that now indicate the infinitesimal rate of execution of the transitions. Within a GSPN, two major types of transitions are possible, *Immediate Transitions* and *Exponential Transitions*.

Immediate transitions, when enabled, will be fired *before* any other enabled transition type. The immediate transition is signified by an exponential density parameter having value ∞. This is in keeping with the mathematical interpretation of the density function. States in the reachability graph of a GSPN still represent all of the states that can be legally reached. If a state is exited due to the firing of an immediate transition, and hence is occupied for zero time, the state is called a *vanishing* state. States which are exited due to the firing of an exponential transition are occupied for some random amount of time and are called *tangible* states. The two differing types of transitions, Immediate and Exponential do not in any way affect the structural analysis of the GSPN. As pointed out by Watson, [Watson 89], [Watson 91a], [Watson 91b], [Watson 92a] and [Watson 92b] the distinction between *vanishing* and *tangible* states does permit a reduction in the size and complexity of the performance analysis problem.

2.4.1.8 Generalized Stochastic Colored Petri Nets

A Generalized Stochastic Colored Petri Net (GSCPN) is an extension to the GSPN and is represented as a sextuplet, where the augmentation is in the introduction of colored tokens to the GSPN construct. Hence a GSCPN, $\Lambda = (P, T, F, \Gamma, C, M^0)$, where $\Gamma: T \rightarrow R_+ \cup \infty$. The colors, C, are used to differentiate between different levels and/or functions of operation occurring within the same GSCPN. Hence through the graphical modeling of a system, multiple levels of operation and/or differing attributes can be viewed simultaneously. During an error recovery, it is possible that more than one level of error-recovery is necessary. Through chromatic identification, differing levels of the error recovery can be distinguished from one another.

2.4.1.9 Inhibitor Arcs

Another extension to the GPN is the use of the *inhibitor* arc. The *inhibitor* arc represents a simplification tool for the graphical representation of GPNs. An *inhibitor* arc placed from a place P_i to a transition T_j will cause the transition to be disabled if the place contains a token. As with normal arcs, a weight can be associated with an inhibitor arc. Hence if a weight, κ, is associated with an arc, then the transition connected to the arc cannot be disabled unless the place connected to the arc contains at least κ tokens.

The inhibitor arc extension to the GPN does not increase the modeling power of the GPN. In fact, it prohibits analysis of the GPN based on the GPN's incidence matrix. The *inhibitor* arc's functionality is easily represented through the use of basic places, transitions and arcs of the GPN. It is a convenience to simplify the graphical representation of the GPN. When considered with an error recovery or on-line plan scheme, the use of inhibitor arcs simplifies the representation of the preconditions needed to execute a plan .

2.4.1.10 Liveness

Liveness in a GPN is qualified by level since a GPN can be hierarchically organized, and is determined by the liveness of the GPN's transitions. Given a Petri Net, Λ, a transition $T_i \in T$ is said to be live under the following criteria:

- Level 0 (dead): $\underline{\chi} \in RS(\Lambda,\underline{\mu}^0) \Rightarrow T_i \notin ES(\Lambda,\underline{\chi})$.
- Level 1 (live): $\exists \underline{\chi} \in RS(\Lambda,\underline{\mu}^0) \ni T_i \in ES(\Lambda,\underline{\chi})$.
- Level 2 (live): given $\kappa \in N_+$, $\exists \sigma \in \Sigma(\Lambda,\underline{\mu}^0) \ni \sigma|_{Ti} \geq \kappa$.
- Level 3 (live): $\exists \sigma \in \Sigma(\Lambda,\underline{\mu}^0) \ni \forall \kappa \in N_+, \sigma|_{Ti} \geq \kappa$.
- Level 4 (live): $\underline{\chi} \in RS(\Lambda,\underline{\mu}^0) \Rightarrow \exists \underline{y} \in RS(\Lambda,\underline{\mu}^0) \ni T_i \in ES(\Lambda,\underline{y})$.

A petri net is said to be live at a level if all of its transitions are live at that level or at a higher level.

2.4.1.11 P and T Invariants

Two types of invariants are defined for GPNs and their extensions. A *P-invariant* of a GPN is a set of places that maintains a constant sum of weighted tokens, throughout the evolution of the net. A *T-invariant* of a GPN is similarly, a set of transitions that, when fired has the aggregate result of leaving the markings unchanged.

2.4.2 Previous Work and Theory on Petri Net Error Recovery

Considerable work has been performed utilizing Petri Nets as the mechanism whereby error recovery of task level errors could be accomplished in an automated manner. There are commonalities among the methods used to accomplish these error recoveries. These will be brought out in the subsections below.

2.4.2.1 Verification of Error Recovery Specification through Colored Petri Nets

Specification of the necessary error recovery parameters can be accomplished at two major intervals in the operation of an intelligent machine. The first interval is prior to the actual call for an error recovery. Establishing a most likely errored event prior to the actual call for an error recovery is a means of identifying a need. Akatsu and coworkers [Akatsu 91] go further in the attempt to identify errored events. They attempt, through the use of Colored Petri Nets, to predetermine a *significant number* of possible error events and verify the actual event against those stored. If no match is obtained, a call to an external agent is made.

This approach, while plausible for an extremely limited size system does pose problems. First, there is no determination as to what constitutes a *significant number* of possible error events. As the distributed system about which they speak is increased in number of distributed processes and complexity of distributed processes, the *significant number* of error recoveries possible must also increase. Utilizing Petri Nets, even colored ones will tend toward a state space explosion of places, arcs and transitions, limiting the actual number of recovery nets that can be stored. Further, it is possible that an error will occur that does not have an error recovery representation. Akatsu does not allow for this possibility, limiting the functionality of his system in an unknown or unstructured environment.

The second interval in which specification of error recovery parameters can be obtained is during an actual error recovery. Akatsu accounts for this type of error recovery specification by limiting his scope of systems to redundant memory or data-plurality systems in which the same data are held in plural or redundant memories. While this approach does provide for the potential of data consistency, a major goal of the Akatsu approach, it does not allow for the majority of systems, which do not have plural memories. In a non-redundant memory system, the Akatsu approach is limited since it is necessary to partition the memory into multiple

independent sub-memories. While this is a possibility, an errant memory manager or memory controller would render the entire memory useless.

2.4.2.2 Adaptive Design of Petri Net Controllers for Automatic Error Recovery

Considerable work has been done by Desrochers, DiCesare, Fielding, Goldbogen, Hörmann and Zhou in the use of Petri Net controllers to provide automatic error recovery. This work is reviewed below.

The basic premise that has been used by the researchers in [Zhou 89], [Fielding 87], [Fielding 88], and [Hörmann 89] is two-fold. First, it is expected by the researchers that the environment in which their system is operating is well known and highly structured. Second, it is expected that given a new error, one that was not anticipated, 'the mechanism of fault diagnosis and error recovery planning will be started by using machine intelligence techniques or operator intervention.....This results in a new Petri Net controller..' [Zhou 89].

Given that the basic premise is sound, which in many cases it may be, the techniques used by the researchers are adequate. The techniques would fall short, however, given a potentially changing or unstructured environment in which new errors would be introduced. These techniques, as taken from [Zhou 89] and [Hörmann 89], are outlined below:

Input Conditioning Method [Zhou 89]

The idea of the input conditioning method is that an abnormal state in a place that represents a process or a state of a manufacturing system can become a normal state after other actions are finished or some conditions are satisfied.

Alternate Plan Method [Zhou 89]

The philosophy of the alternate plan method (also called error avoidance) states that there exists another Petri net controller that can transform an abnormal state in place *p* directly into a normal state in the same Petri Net.

Backward Error Recovery Method [Zhou 89]

Backward error recovery suggests that under the assumption that the state is normal (e(p)=1), a Petri Net controller, *S*, is executed and a new faulty state in place *w* results. But this state can become a normal state in place *p* after the operation of a Petri net controller *S'*.

Forward Error Recovery Method [Zhou 89]

The forward error recovery method is similar to the backward error recovery method. Suppose that a faulty state results after the operation of a Petri Net controller, *S*. However, this state can be directly transformed into a normal state in after a Petri net controller *S'* operates.

As stated by Zhou, the methods he presents would be particularly effective if 'an easy way to modify the control code when a new error is encountered and the procedures to handle it are determined,' were available. It is here that the methods outlined by Zhou and coworkers fall short. There is no guarantee that control code modifiers will be available. Hence Zhou's methods are relegated to systems that are completely defined and limited in the complexity of the potential problems that could be handled. In addition, Zhou's work deals only with Ordinary Petri Nets. He does not consider the use of Generalized Stochastic Petri Nets or Generalized Stochastic Colored Petri Nets that introduce exponentially timed transitions.

Hörmann utilizes a failure monitor to establish fault diagnosis but does not expand on the interconnection between the failure monitor and the error recovery schemes that would be generated. Although he does utilize condition / event Petri nets for sequencing tasks as well as a world model for storing information about the current activities he shows no connection between

the world model and the generation of other than preestablished activities. As with Zhou, he expects the environment to be highly structured.

2.4.2.3 Summary

Utilizing a purely Petri Net approach in the attempt to create a robust error recovery scheme has been shown to work only in extremely limited systems. Those systems exist in highly structured well known, non-changing environments. The introduction of a changing environment adds the potential for new information which must be transformed into useful Petri Net places, transitions and arcs. The techniques reviewed do not provide for the capability to accomplish this. In addition, the modeling of a changing environment with a Petri Net does not account for the potential of state-space explosion, a very real problem.

2.5 Semantic Networks

This section describes the Semantic Network (*SNet*) structure introduced by Quillian [Quillian 68] and previous work using Semantic Networks, applicable to this Thesis Proposal. This is the first time that *SNets* have been proposed for error recovery or on-line planning.

2.5.1 Semantic Network Structure

As defined in [Quillian 68], [Nilsson 80], [MacRandal 88], and [Shastri 88], a Semantic Network is an abstract conceptual structure representing knowledge in terms of concepts, their properties, and the hierarchical sub/superclass relationship(s) that exists between concepts. It presents the knowledge as nodes, representing conceptual units, and directed links representing the relationships between units, in a net-like graph. The essential idea behind *SNets* is that this

graph theoretic structure of relations and abstractions can be used not only for inference, but also for understanding.

Unlike specialized networks and other graph theoretic structures such as Petri Nets, *SNets* aim to represent any kind of knowledge that can be described in natural language. In addition, the Semantic Network provides methods for automatically deriving larger bodies of implied knowledge without destroying the underlying body of knowledge explicitly stored in the Semantic Network structure. This approach remains valid even for complex structures, since any non-atomic structure can be shown to have some composite structure that can be decomposed for storage, provided that its characteristic relations are maintained.

Semantic Networks possess multiple layers of abstraction that permit the Semantic Network to maintain multiple classes and superclasses for state description. This capability is of importance in the modeling of hierarchical structures in which purely mathematical modeling is ineffective. Activities, such as linguistic analysis, which fall into this category include those in which conceptual analysis is required as opposed to the repeated processing of modeled elements. A Semantic Network provides a map of the semantic meaning of a natural language sentence, in an ordered, arranged, structured knowledge base. This permits syntactically different sentences to be immediately related by their semantic meanings. Some previous work with Semantic Networks is applicable to this Thesis Proposal. The following subsection examines some of this work.

2.5.2 Previous Work Using Semantic Networks

A Semantic Network has been shown to be a viable construct for the development of databases, both in terms of speed of access, and minimization of space. Richens [Richens 56a] [Richens 56b] uses a core of semantic primitives created on a Semantic Network for language translation.

His creation, **Nude**, utilizes this core of semantic primitives to build other related concepts. His work was furthered by Masterman [Masterman 62] in her **T-Lattice** construct, used for the generation of a thesaurus for organizing language concepts hierarchically. These two pieces of work are important as they establish the successful use of primitive cores of concepts from which more complex concepts can be built.

This importance of the concept core is not the only important development in the use of Semantic Networks. It was realized by Brachman [Brachman 85] that in addition to the general concepts that form the Semantic Network nodes, specific information is necessary to ensure the distinction of objects that fall within the same conceptual classes. His work resulted in the establishment of five link/node levels representing a Semantic Network description. These levels are depicted in *Figure 1*, from highest level to lowest level. Brachman showed that a description using a Semantic Network could exist on all of the levels simultaneously, with objects and relations at one level being realized using the structures of a lower level.

The difficulty is in representing structures in a computer environment. This difficulty was initially eliminated through the use of the *frame* construct. However, Fillmore [Fillmore 86] and Simmons [Simmons 73] showed that the *frame* construct is insufficient. They postulated that the semantic case represented the real world role played by an actor, in an event. Hence they applied restrictions to Minsky's frames and created the *case_frame* structure. This new frame type is characterized by an event, its cases and the type restrictions placed on related objects.

Level	Component	Structure
Linguistic	Arbitrary Concepts Words, Expressions	Sentence Descriptions
Conceptual	Semantic or Conceptual Relations (cases) Primitive objects	Concept Dependencies Deep Case semantic networks
Epistemological	Concept Types Inheritance, etc.	Associative Relational
Logical	Propositions Predicates Logical Operators	Boolean Logic Nodes
Implementation	Atoms Pointers	Data Structures Frames

Figure 1
Brachman's Analysis

The applicability of the *case_frame* construct to this Thesis Proposal is high due to the fact that actions are limited by the environment in which the actions take place. Even were the environment to be unstructured, the relationships between objects within the environment still maintain a structure.

Important to the development of this Thesis Proposal is the ability to search a Semantic Network for paths between object nodes, effectively determining the connectivity between the nodes. Work performed by Hendler, Quillian, Norvig, Yu and Riesbeck, [Hendler 92] [Quillian 69] [Norvig 89] [Yu 90] [Riesbeck 86], has shown that symbolic *marker passing* is a viable means of deriving a path between two Semantic Network nodes. Symbolic marker passing is a technique which initially identifies two nodes as nodes of interest (origins), and then proceeds to identify (mark) all neighboring nodes until a node is marked from the two differing origins. The

potential for the state-space explosion of marked nodes and false path identification is eliminated through restrictions on type, and limitations on acceptable link traversals.

2.5.3 Summary

Although there is no work that has been done using Semantic Networks for error recovery and on-line planning, there has been extensive work done using Semantic Networks for other purposes. This Thesis Proposal proposes the use of Semantic Networks in the establishment of the Primitive Structure Database of the Planning Coordinator, an essential part of the Planning Coordinator's error recovery and on-line planning scheme. The Planning Coordinator is the subject of the next section.

3.0 The Planning Coordinator

The Planning Coordinator is a stand alone error recovery architecture designed to be used with an intelligent *x-autonomous* system architecture. Because it is expected to be a stand alone architecture, the Planning Coordinator's relationship to any existing *x-autonomous* architecture is as a logical extension, functionally subservient to the main controller of the *x-autonomous system*. This is not to imply that the Planning Coordinator cannot function on its own. Having reviewed several distinct system architectures, as outlined in Section 2, those that correspond to the principle of Increasing Precision With Decreasing Intelligence have shown themselves to be most promising to the attachment of the Planning Coordinator as an error recovery unit. In particular, the Intelligent Machine Model (IMM) developed by Saridis and coworkers has demonstrated the most flexibility. Due to its flexibility, the Intelligent Machine Model has been chosen as the intelligent architecture which the Planning Coordinator will use for identification and demonstration of its capabilities.

The following subsection briefly examines the Intelligent Machine Model and those factors that have made it the choice for the initial application of the Planning Coordinator. Following this subsection will be an examination of the Planning Coordinator's Macro Structure, Interface Description, and Functional Description.

3.1 Intelligent Machine Model

The Intelligent Machine Model, in its present form, is based upon the theory of hierarchically intelligent controls introduced by Saridis and Stephanou [Saridis 77] and furthered by Saridis through the analytic design of Intelligent Machines [Saridis 83]. The theory of hierarchically intelligent controls proposes the stratification of the Intelligent Machine into three levels based upon the theory of Increasing Precision with Decreasing Intelligence.

The three levels depicted in *Figure 2*, the Organization Level, the Coordination Level, and the Execution Level, function conceptually to provide high level decision making and long term planning, decision making and short term learning, and control function execution, respectively.

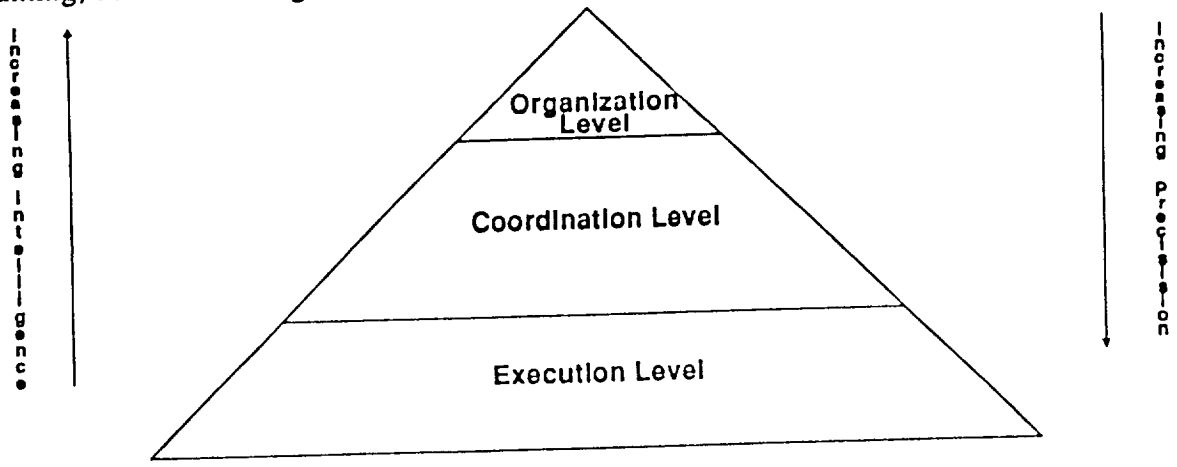


Figure 2
THE INTELLIGENT MACHINE MODEL

As expanded in *Figure 3*, the Organization Level, consists of an Organizer and a Global World Model. The Organizer, functions solely to interpret the Global World Model and thus develop off-line, long term strategic plans. These off-line, long term plans function collectively to provide the Intelligent Machine with long term goals. Individually, these off-line, long term plans function to provide the Intelligent Machine with medium term goals that serve to achieve milestones along the long term path. The Global World Model is a quasi-static representation of both the present environment in which the Intelligent Machine must function, and the past environments in which the Intelligent Machine has functioned. For the purposes of this Thesis Proposal, it is necessarily assumed that the Organization Level in its entirety is completely functional.

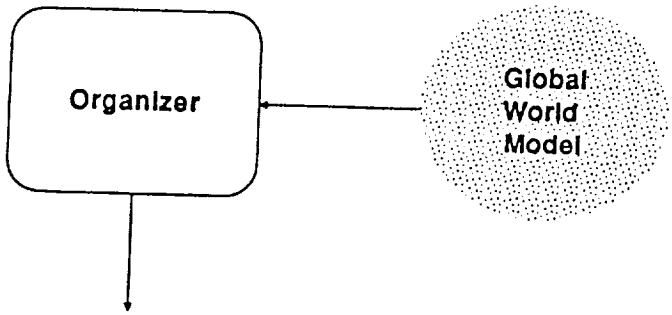


Figure 3
The Organization Level

As expanded in *Figure 4*, the Execution Level functions to take the detailed commands sent to it from the various Coordinators of the Coordination Level and execute them on the physical hardware that is part of the Intelligent Machine. Note that this hardware is not limited to robotic elements but can be any tangible piece of equipment that can perform some *physical* or *mental* function. As regards this Thesis Proposal, it is assumed that the Execution Level and the hardware it must interface with are completely functional.

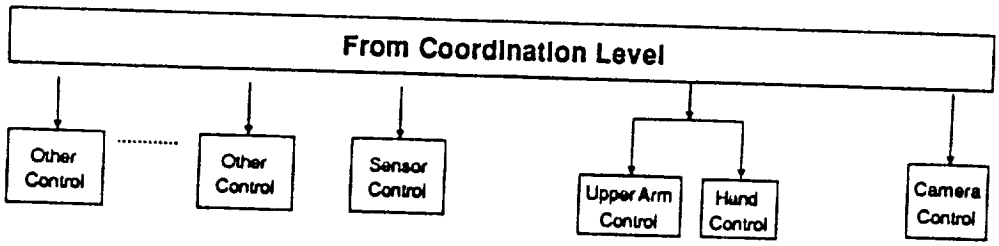


Figure 4
The Execution Level

Of interest to this Thesis Proposal is the Coordination Level, the logical point of interface for the Planning Coordinator. As originally designed by Saridis [Saridis 88], the Coordination Level consisted of a Dispatcher that interprets the plans sent from the Organization Level, and

Coordinators that function to interpret subplans sent from the Dispatcher and convert them into control commands understandable by the physical hardware whose actions the Coordinators coordinate. Depicted in *Figure 5* is a modified Coordination Level. The modified Coordination Level is very similar to that of the original Coordination Level. The major difference between the two is in the inclusion of Coordinators that do not directly connect to the physical hardware of the Execution Level. Among these Coordinators are those that are physically part of the Intelligent Machine, such as the Learning Coordinator, and those that are logical extensions to the Intelligent Machine, such as the Planning Coordinator.

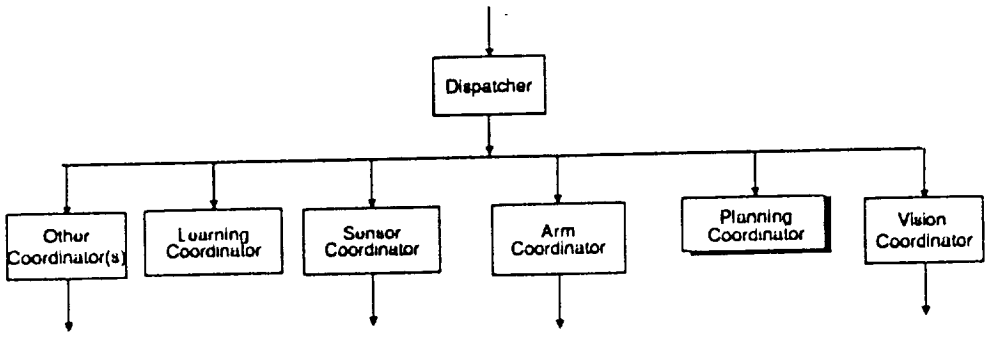


Figure 5
The Coordination Level

The complete architecture is depicted in *Figure 6* on the following page. From the figure, it can be seen that the Planning Coordinator attempts to take advantage of existing modules and existing communication pathways. By doing so, the Planning Coordinator architecture becomes a more versatile architecture.

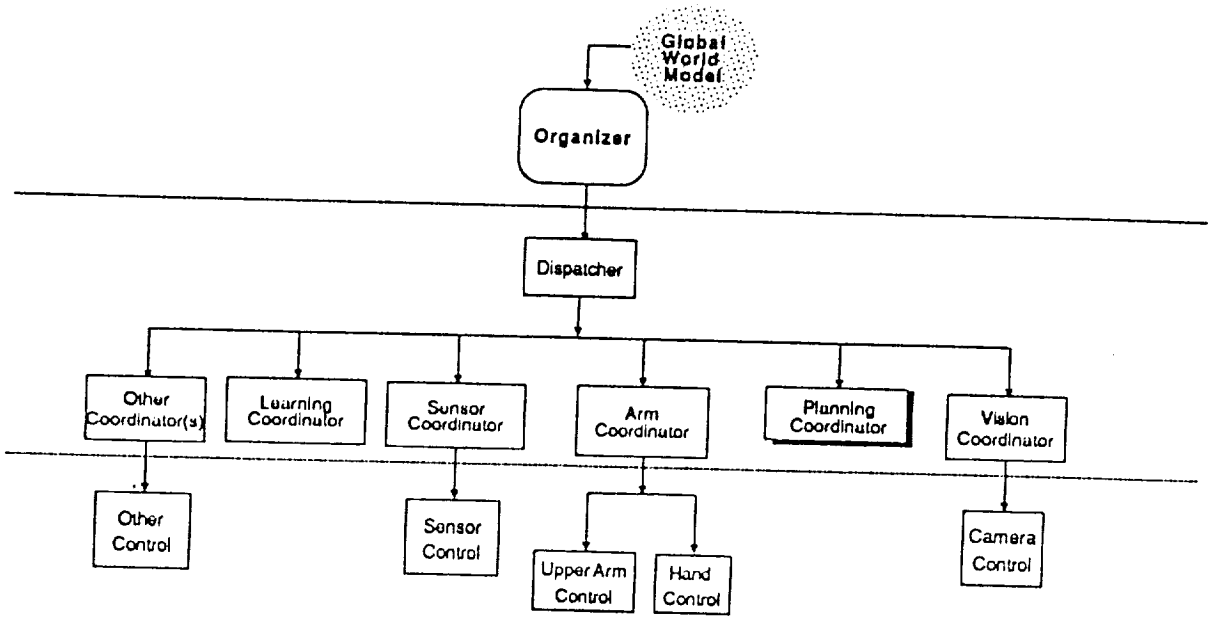


Figure 6
The Levels of the Intelligent Machine Including the Planning Coordinator

Except where needed for clarification, the other Coordinators in the Coordination Level will not be discussed. The following subsections examine the Planning Coordinator.

3.2 Planning Coordinator: Macro Architecture

Physically, the Planning Coordinator is a functionally stratified, stand alone device operating as a logical extension to the Coordination Level of the Intelligent Machine. It logically connects to the physical communication scheme of the Intelligent Machine through external communication ports. Unlike the other Coordinators in the Intelligent Machine, the Planning Coordinator does not communicate directly to any piece of physical hardware or to the Organizer. To engage in

communication, the Planning Coordinator utilizes the communication schemes that exist between the other Coordinators and the hardware they coordinate, as well as the communication scheme between the Dispatcher and the Organizer.

As a logical extension to the Intelligent Machine, the Planning Coordinator is logically subservient to whatever is considered to be the main controller of the Intelligent Machine. This is necessary to ensure system coherence. In the event of a catastrophic main controller failure the Planning Coordinator might assume the role of the overall system controller, but to a very limited extent. The Planning Coordinator is expanded in *Figure 7*, to introduce the constituent parts of its macro architecture, listed below. These parts are grouped by level.

Current World Model (CWM) - Level 1
Shadow Coordination Level Petri Net (SCPN) - Level 1
Primitive Structure Database (PSDB) - Level 2
Node/Link Weighting Mechanism (N/L-WM) - Level 2
Mapping Mechanism (MM) - Level 2
Error Recovery Generation Algorithm (ERGA) - Level 2
System Fault Monitor (SFM) - Level 3

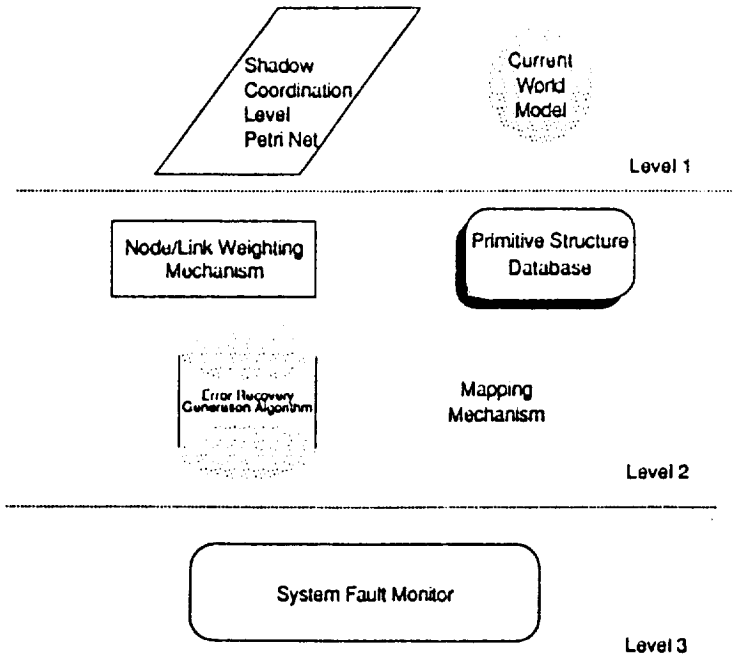


Figure 7
The Planning Coordinator (PCOORD) Constituent Parts

With the exception of the System Fault Monitor, these components constitute the mechanisms whereby a task level error recovery (or on-line plan which henceforth is considered to be a specific instantiation of an error recovery), will be successfully executed. The following subsections elaborate on the seven constituent parts that comprise the Planning Coordinator. Of highest significance to this Thesis Proposal are the five major component parts of the Planning Coordinator: *Current World Model, Primitive Structure Database, Node/Link Weighting Mechanism , Mapping Mechanism,* and *Error Recovery Generation Algorithm.*

3.2.1 The Current World Model

The Current World Model is a dynamically changing, linguistic representation of the most current environmental information available to the Planning Coordinator. Its function is to accurately represent the most current status of the environment in which the Planning Coordinator, and hence the Intelligent Machine, must operate. Unlike the long term, quasi-static Global World Model, the Current World Model maintains a shorter term representation. With two major exceptions, only a portion of the Current World Model will be active at any one time. The first exception occurs when information from the Current World Model is initially interpreted to create the Primitive Structure Database. The second exception occurs when the Current World Model is called upon to update the Global World Model with new information derived from the activities of the Planning Coordinator.

Note that in the development of the Planning Coordinator, it is anticipated that the Current World Model will change significantly over a long period of time and as such, differ significantly from the Global World Model. A question arises as to the need for the Current World Model if a Global World Model exists and can be made to be accessed reliably and efficiently. This question is answered as follows. The Primitive Structure Database, to be

discussed, represents Primitive Structures derived from the Current World Model. When the Current World Model changes, the existing Primitive Structures are not lost. They remain in the Primitive Structure Database which is augmented through the addition of the new primitive structures. Hence were the Global World Model to succumb to a catastrophic failure, none of the information that had been contained in it would be lost. This is because the Global World Model can be regenerated from the information stored in the Current World Model and the information stored in the Primitive Structure Database. Thus it is necessary that the active portion of the Current World Model maintain coherence with the corresponding portion of the Global World Model. This coherence is ensured through communication and update between the Current World Model and Global World Model. It is more important that as object and event primitives are generated, that they have representation in both the Primitive Structure Database and the Current World Model. This coherence will facilitate the regeneration of the Global World Model from the Current World Model and the Primitive Structure Database. This regeneration is not of concern to this Thesis Proposal. It is considered to be potentially future work outside of the scope of this Thesis, and is of itself a research area.

3.2.2 Shadow Coordination Level Petri Net

Within the hierarchy of the Intelligent Machine Model, task representation is performed through the use of Generalized Stochastic Colored Petri Nets as described previously. To maintain the continuity of this representation during its design phase, the Planning Coordinator utilizes a Shadow Coordination Level Petri Net, as a graphical representation tool. It is anticipated that the Shadow Coordination Level Petri Net will eventually become unnecessary and will be eliminated as a graphical representation tool. However, its function will be maintained.

The Shadow Coordination Level Petri Net is functionally, an exact copy of the executing Coordination Level Petri Net generated by the Dispatcher of the Coordination Level of the

Intelligent Machine. Depicted in *Figure 8A* is a Coordination Level Petri Net, and in *Figure 8B*, the Shadow Coordination Level Petri Net, identified by its transitions. These transitions maintain connections to Map Interface Error Recovery Nodes that reside within the Mapping Mechanism of the Planning Coordinator. Through these connections it is possible to determine the exact location from which an error recovery would be enacted should the need for one arise, since errors occur only at Petri Net transitions. Identifying the locations of potential errors permits the pre-creation of most likely errored event recovery plans. Hence when an error does occur, and is the most likely errored event, an immediate response is possible. Considering an error that is not the most likely errored event requires the use of alternate plans. These alternate plans are built up from Primitive Structures that represent the actions and objects existing in the environment of an intelligent machine. They can be stored in a database for retrieval. This database is called the Primitive Structure Database and is the subject of the next subsection.

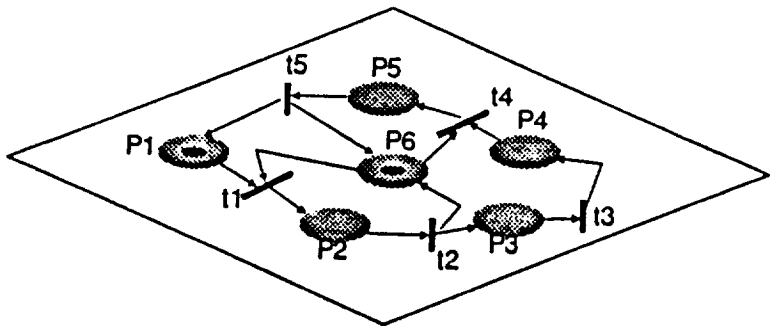


Figure 8A
A Coordination Level Petri Net

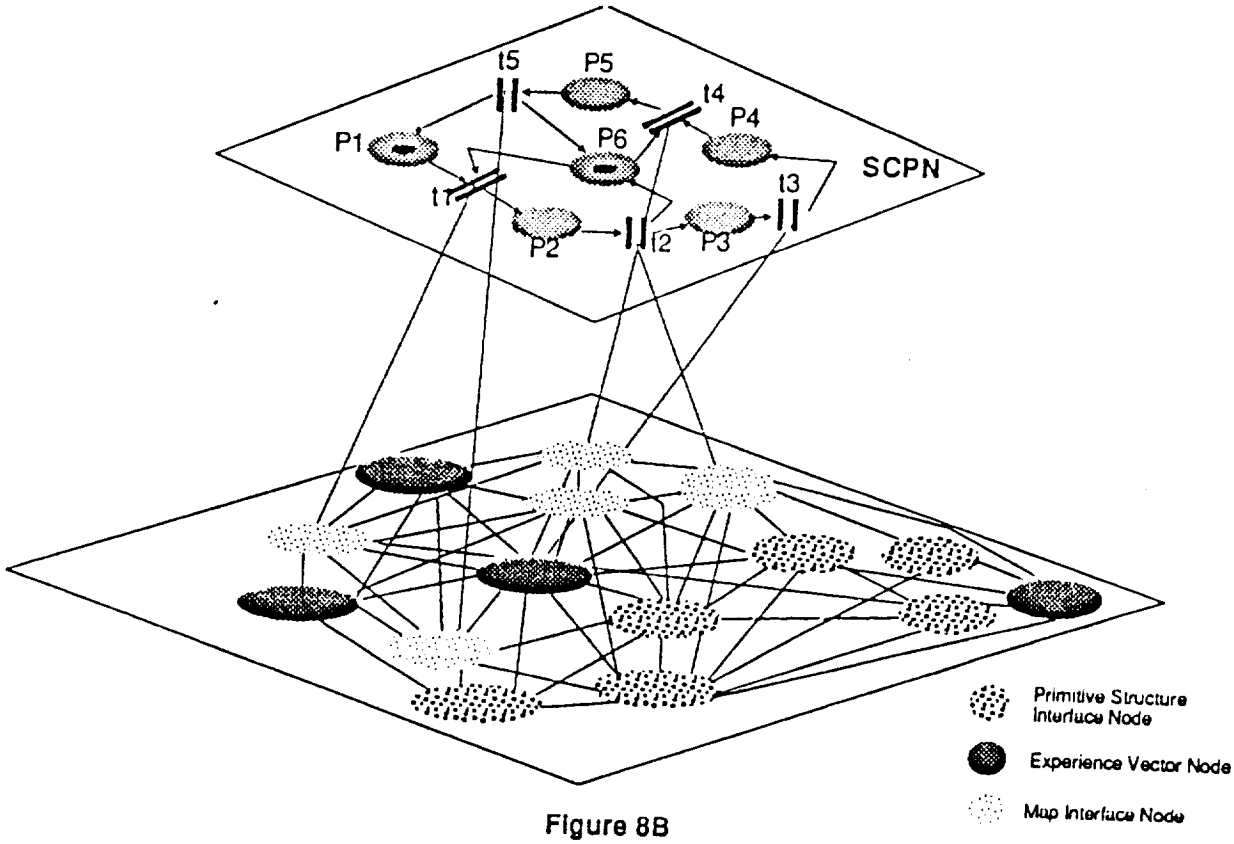


Figure 8B

The Shadow Coordination Level Petri Net Equivalent

3.2.3 The Primitive Structure Database

The Primitive Structure Database (PSDB) is a database containing Primitive Structures that represent the basic operations that can be performed by an intelligent machine, as well as the objects that exist in the environment of an intelligent machine. These Primitive Structures are derived from the Current World Model which represents the most up to date environmental information available to the intelligent machine. The formal definition of a Primitive Structure is given below:

> Primitive Structure
>
> A potentially complex, building block created by the Planning Coordinator, based upon environmental information contained in the Current World Model, and functioning to represent the primitive actions (objects) capable of being performed (identified) by the intelligent machine. Several Primitive Structures can be combined to form complex plans that can later be used as either error recovery plans or on-line plans.

In keeping with the general structure of the Intelligent Machine Model, the Primitive Structures are individually, live, safe, bounded Petri Nets. Synthesizing these smaller Petri Nets into larger ones has been shown by Zhou, DiCesare, Narahari, and Koh to result in live, safe, bounded Petri Nets given that the properties of liveness, safeness, and boundedness existed in *each* of the smaller Petri Nets [Zhou 88] [DiCesare 88] [Narahari 88] [Koh 88]. In *Section 2*, both Generalized Stochastic Colored Petri Nets and Semantic Networks were introduced. The following builds on the descriptions of these two constructs.

The Primitive Structure Database is modeled using a Semantic Network. The Semantic Network model makes use of nodes, representing events and/or objects, and directed arcs, representing the

relationships between objects. The nodes can be hierarchically structured, thus permitting descendants to inherit form and function from their ancestors. This is important because the descendants themselves may be separate nodes in the *PSDB*. In addition, through the use of *case_frames* as previously described, inherent search limiting agents are built into the Primitive Structure Database nodes. Finally, each of these nodes is connected in some way to other nodes. The connections may be simple or multiple, depending upon the complexity of the Primitive Structure Database. These connections are achieved through the use of linguistically identified, directed, relational arcs. The relational arcs permit conceptual relations between the nodes and as such are the natural points at which the strengths of such relations can be established. Since the relational links are directed, they provide natural pathways from one node to another. These natural pathways can be exploited to establish ordered error recovery or on-line plans. There is a difficulty in using this approach, however. There may be multiple paths between any two nodes in a network. To assign a strength value to each of the links between any two nodes and to distinguish between the multiple paths that may exist between any two nodes (i.e., choose the best path from among all), the Node/Link Weighting Mechanism was introduced. The Node/Link Weighting Mechanism, is the subject of the next subsection.

3.2.4 The Node/Link Weighting Mechanism

The Node/Link Weighting Mechanism is one of the five major components of the Planning Coordinator. It functions to assign fuzzy weights (*f-weights*) to the nodes and relational links that comprise the *SNet* based Primitive Structure Database. The *f-weights* are used for two purposes. The primary purpose is to establish the relational strength(s) of one node to another, based on the linguistic relation connecting them. The secondary purpose is to combine the *f-weights* assigned to each individual relational link in some established path plan, and defuzzify the result. The defuzzified result provides the overall possibility of success number that the plan's path represents. The possibility of success number is then used to hierarchically organize,

from highest possibility of success to lowest possibility of success, all of the plan paths whose possibility of success numbers' exceed some established threshold value. This organized list, called a plan execution list, contains those plans that have been deemed applicable to some error recovery or on-line plan request.

Once a possibility of success number has been determined, each of the relational links along the plan path that is responsible for the number's creation is assigned an ordering identifier, indicating which plan or plans in the execution list, the link refers to. The assignment of the relational *f-weights*, the combining of a series of relational *f-weights* into an overall plan *f-weight*, and the defuzzification of the overall plan *f-weight* into a 'crisp number' are examined in the following subsections.

3.2.4.1 Assignment of Relational F-Weights

The universe of discourse represented in an intelligent autonomous system is that derived from the system's knowledge of its environment as transformed into Primitive Structures and the relationships between Primitive Structures. As has been stated, the Primitive Structures are maintained in a non-fedback *SNet* which is a structure very similar to the Fuzzy Cognitive Map of Kosko and Styblinski [Kosko 86] [Styblinski 88]. These similarities permit the use of Fuzzy Cognitive Map techniques.

In establishing the Primitive Structure Database the first step is to derive the Primitive Structures and the relationships between Primitive Structures from the environmental information. Here it can be assumed that the Primitive Structures have been made available with no loss of generality. The resulting universe of discourse is an arbitrarily large but finite set of interconnected nodes, similar in structure to the Dempster-Shafer frame of discernment [Kosko 87]. The major difference results from the fact that the

universe of discourse is dynamic and hence it is necessary to use a Semantic Network base which permits changes in the base without destruction of the existing base.

The Node/Link Weighting Mechanism utilizes a dynamic Fuzzy Rule Base created by an Expert System from the available environmental information. Prior to the operation of an *x-autonomous* system, there are basic rules available, akin to the instinctual capabilities that human beings possess from birth. As the *x-autonomous* system begins to operate, its environment changes and the Expert System derives new rules to be added to the dynamic Fuzzy Rule Base. The feasibility of the dynamic Fuzzy Rule Base has been demonstrated [Kosko 86]. The Primitive Structure Database, is initially a semantic network. Until application of the Node/Link Weighting Mechanism, there are no *f-weight* relations. The Node/Link Weighting Mechanism takes two connected nodes in the Primitive Structure Database and their linguistic relational arc and applies them to the Fuzzy Rule Base. The result of applying these two nodes and the arc is an *f-weight*, which is applied to the arc. When two nodes have multiple connections, potentially differing *f-weights* are assigned to each of the individual relational arcs. In this way, two nodes can have varying degrees of relational strength, based on the relation itself. This same procedure is applied to all pairwise nodes in the Primitive Structure Database according to the general procedure outlined below.

General Procedure

(Prior to beginning x-autonomous system operation)

Step 1: From information in the Current World Model, use Expert System to derive dynamic Fuzzy Rule Base.

Step 2: From Current World Model derive the nodes and links for primary semantic network.

Step 3: Beginning from any node in the resulting connected digraph, utilize minimal time, complete search techniques to search and mark the entire digraph. During marking, take any two connected nodes and the directed arc between them and apply them to the Fuzzy Rule Base, resulting in an arc *f-weight.*

Step 4: Apply the *f-weight* to the relational link and return to *Step 3* until the entire digraph is done.

(Upon completion of the Node / Link Weighting)

Step 5: As new nodes are added to the newly created Primitive Structure Database, begin at a newly introduced node and determine which node or nodes it is connected to. Apply the newly introduced node and the nodes it is connected to, to the Fuzzy Rule Base and determine an *f-weight.* Apply the *f-weight* to the new relational link as before.

End Procedure

Utilizing the above permits the establishing of a new Primitive Structure Database, or the augmenting of an existing Primitive Structure Database. As has been described, it is possible to start at one node (i.e., a start node) and efficiently trace out a path or paths to another node (i.e., a destination node). It is likely that with the high probability of multiple connections existing between two nodes, there will be multiple paths between two nodes. Within the confines of error recovery it is necessary to differentiate these paths into a hierarchically ordered plan execution list. This requires the combining of individual link *f-weights* into an overall plan *f-weight*, the subject of the next subsection.

3.2.4.2 Determining Overall Plan F-Weight and Creating Plan Execution List

Once the Primitive Structure Database has been constructed it is ready to be used by the Planning Coordinator for error recovery. In its final form, the Primitive Structure Database resembles a Fuzzy Cognitive Map. This resemblance permits the application of Fuzzy Cognitive Map Summation Methods to sum the individual link weights along a particular path. Once the weights along a particular path have been summed, the overall value is defuzzified into a crisp number. This crisp number is then used for comparison against a threshold value. If the number exceeds the threshold value, the plan is considered viable and is placed in the plan execution list. The general procedure is given below:

Plan Execution List Generation

Step 1: Identify plans generated by Primitive Structure Database search.

Step 2: For each plan, start at the start node and trace plan through to the destination node. At each connecting link, assign plan identifier to each link along the plan path, and store each link fuzzy value.

Step 3: For each traced through plan, take link fuzzy values and apply Fuzzy Cognitive Map Summation Method to it. Defuzzify the result into a crisp number. If the crisp number represents a value that exceeds the established threshold value, store the value and the plan identifier in the Plan Execution List, else discard it.

Note: If crisp number represents first plan, store in first slot of Plan Execution List regardless of its value. This will ensure that at least one plan is available to be tried. For each subsequent plan entered into the Plan Execution List, use binary search to find the correct position for the new plan in the Plan Execution List. If subsequent plan values exceed the first plan value, but do not exceed the threshold, then replace the first plan with the subsequent plan.

Step 4: Update link identifiers in Step 2, to reflect plan position in the Plan Execution List of Step 3.

Note: It is possible that a single link may be needed for more than one plan. Due to this possibility, a single link may have multiple identifiers.

End Procedure

The result of this general procedure is a Plan Execution List containing ordered viable plans, with ordering numbers. These ordering numbers are used to execute the plans sequentially until one plan is successful or until all plans have been exhausted. If a plan is executed and is successful, the plan is rewarded. If a plan is unsuccessful, it is penalized. The method of applying a reward or penalty is as yet undetermined and remains an open area for further research. Previously, it was stated that through the Shadow Coordination Level Petri Net it is possible to immediately identify where an error recovery must begin and end. This is due to the fact that errors can occur only at the Petri Net transitions. These transitions are connected to Map Interface Nodes that reside on *Level 2* of the Planning Coordinator. The following section introduces the Map Interface Nodes as well as the Mapping Mechanism of the Planning Coordinator.

3.2.5 Mapping Mechanism

The desired one-to-one mapping mechanism of the Planning Coordinator is both a structure to maintain three specific node types and a methodology to efficiently perform the following three functions:

1. Connect Shadow Coordination Level Petri Net transitions to *Map Interface Nodes*.

2. Connect *Primitive Structure Interface Nodes* to their Primitive Structures in the Primitive Structure Database.
3. Create *Experience Vector Nodes* based upon previously enacted, successful error recoveries.

The three different node types are defined below.

Map Interface Node

A dynamically allocated, two or three port active, interface point that connects to a Shadow Coordination Level Petri Net transition, the start (end) node of an error recovery or on-line plan, and an Experience Vector Node. Two Map Interface Nodes are created for each Shadow Coordination Level Petri Net transition. They are created at the same time as the Shadow Coordination Level Petri Net.

Primitive Structure Interface Node

A Primitive Structure Interface Node represents a pointer to a Primitive Structure in the Primitive Structure Database. It is necessary that each Primitive Structure be represented by a Primitive Structure Interface Node to ensure the identification of the start and end nodes of an error recovery or on-line plan.

Experience Vector Node

An Experience Vector Node is attached to the third port of the input Map Interface Node (i.e., the side connected to the input side of the Shadow Coordination Level Petri Net transition). It represents a successfully enacted error recovery or on-line plan sequence. The Experience Vector Node maintains the entire plan path through a vector of Primitive Structure Interface Node identifiers. These identifiers maintain the order of execution of the nodes. In addition, the Experience Vector Node maintains the state of the system when the error occurred. This permits an immediate response to an *identical* error.

Note that if the same error recovery does not work for an identical *error, the Experience Vector Node is updated with information on the new error solution, when the new error solution is found.*

In order that a Map Interface Node can connect to any Primitive Structure Interface Node, as is required for the operation of the Planning Coordinator, the network of Map Interface Nodes and Primitive Structure Interface Nodes must be a fully connected network. The choice of which links in the Map Interface Node / Primitive Structure Interface Node network to make active and which to leave inactive is determined by the places in the Shadow Coordination Level Petri Net that immediately precede and follow a transition connected to a Map Interface Node.

A considerable amount of the work that the Planning Coordinator must do can be done while the overall Intelligent Machine is not engaged in error recovery. As a result, the preprocessing of the error recovery plans can be done in parallel with the execution of the Intelligent Machine, saving overall execution time. This does not mean that the Planning Coordinator's primary function as an error recovery unit remains dormant until the preprocessing is done. The Planning Coordinator's overall function is governed by the Error Recovery Generation Algorithm, the subject of the next subsection.

3.2.6 Error Recovery Generation Algorithm

The Error Recovery Generation Algorithm governs the operation of the Planning Coordinator. The algorithm itself assumes the availability of specific information from the Intelligent Machine, the accessibility of communication ports to the Intelligent Machine and priority over all other coordinators. Some of the information which must be provided by the Intelligent Machine to the Planning Coordinator includes.

- Error status based upon a flag called ERR_FLAG. If asserted, an error is present. If not asserted, no error is present.

- On-line Plan status based upon a flag called OP_FLAG. If asserted without ERR_FLAG, a *short term* on-line plan is required. If asserted with ERR_FLAG, an *interactive* on-line plan is required.
- Intelligent Machine main controller status based upon a flag called MC_FAIL. If asserted, the main controller has failed.

Figure 9 shows the two stage flow diagram of the operation of the Error Recovery Generation Algorithm. Stage 1, details the preprocessing of the initial Current World Model, Fuzzy Rule Base, and Primitive Structure Database. Stage 2, details the preprocessing of the initial error recovery routines and the processing of error recoveries, on-line plans, and modifications to the Primitive Structure Database. The steps involved in the Error Recovery Generation Algorithm are detailed following the figure.

Prior To Commencing Operation of the Intelligent Machine

Step 1

From the Global World Model create the Current World Model. Initially, the Current World Model and the Global World Model will be the same.

Step 2

A) From the Current World Model, use an Expert System to create the dynamic Fuzzy Rule Base. Initially this base may contain rules that are considered to be instinctual.

B) From the Current World Model, derive the underlying Semantic Network of the Primitive Structure Database. This includes both nodes and links.

C) For each node created for the underlying Semantic Network of the Primitive Structure Database, create a Primitive Structure interface node.

Step 3

From the Fuzzy Rule Base and the Semantic Network create the Primitive Structure Database by marking the SNet and applying each pairwise connected set of nodes and their connecting link to the Fuzzy Rule Base, yielding an *f-weight.*

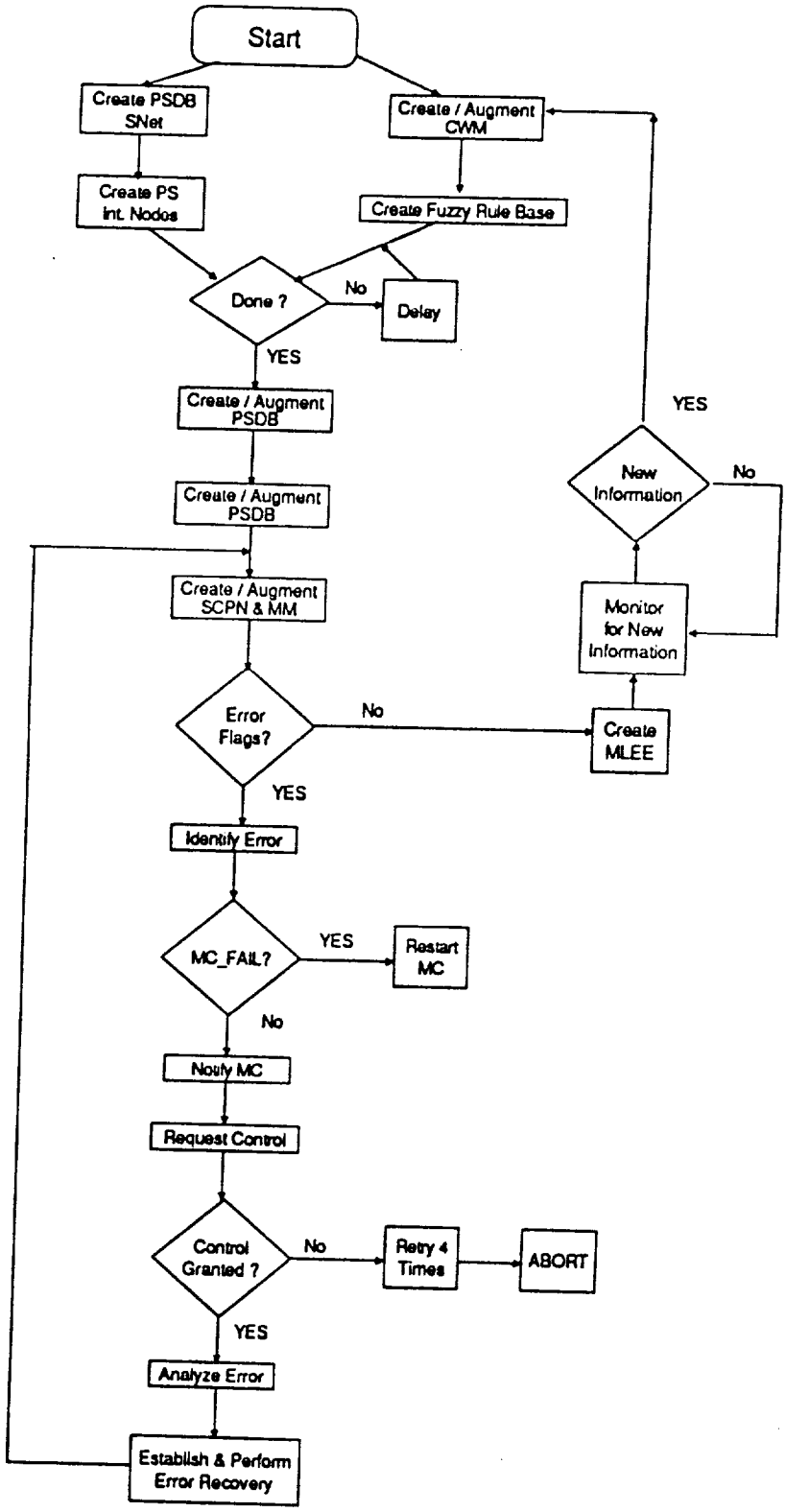


Figure 9
Functional Flow Diagram

After Commencing Operation of the Intelligent Machine

Step 4

Create Shadow Coordination Level Petri Net and establish connections to Map Interface Nodes.

While neither ERR_FLAG, nor OP_FLAG nor MC_FAIL is asserted perform Step 5 else perform Step 6:

Step 5

A) Preprocess most likely errored event for each Shadow Coordination Level Petri Net transition and store resulting plan.

B) Monitor introduction of new information into the Current World Model. If applicable perform Step 2 and augment Primitive Structure Database as per Step 3.

Step 6

Using ERR_FLAG, OP_FLAG and MC_FAIL, attempt to identify the error from the information given by the Intelligent Machine. For ERR_FLAG and OP_FLAG, determine the Shadow Coordination Level Petri Net transition from which the error began, and activate Map Interface Node. For MC_FAIL , assume system control and attempt system restoration. If successful, return system control to the Intelligent Machine main controller. If unsuccessful, notify external base for assistance and gracefully shutdown Intelligent Machine operation.

Note: There are several types of errors possible during task execution. These error types are outlined in Subsection 3.4.1.

Step 7

Notify the Intelligent Machine's main controller that an error has occurred.
Note that the main controller may already know that an error has occurred. This step is to ensure the continuity of error data transmission.

Step 8

A) Request control of the Intelligent Machine's operation from the main controller to prevent interference during error recovery. If not granted, re-try four times. If not granted, abort error recovery.

B) If granted, analyze error information from Step 6. If information is sufficient to enact error recovery, do so. Otherwise use system components (i.e., Vision System, Motion Control System etc.) to try to gain further information on the error type.

Step 9

A) If an error is the same as the calculated most likely error, then execute the preprocessed most likely error, error recovery. If the error is not the most likely error, then establish recovery plans in the plan execution list.

B) Execute first plan in the plan execution list. If successful return control to main controller. If unsuccessful, execute remaining plans in the plan execution list until either all plans are executed or one is successful. If no plans are successful, report irrecoverable error to the main controller and return control to the main controller.

Step 10

Return to Step 5 and continue.

This concludes the Error Recovery Generation Algorithm. The next subsection outlines the System Fault Monitor.

3.2.7 System Fault Monitor

The System Fault Monitor functions to monitor and perform hardware diagnostics of the Planning Coordinator and if desired, the Intelligent Machine to which the Planning Coordinator is connected. Although it performs an error recovery function for hardware, the System Fault Monitor is not one of the major components of the Planning Coordinator. This is because its

function is in an area that has been very highly developed. Existing fault diagnostic techniques do suffice. As such, the further development of the System Fault Monitor will not be considered for this Thesis Proposal.

3.2.8 Summary

The Macro Architecture of the Planning Coordinator consists of seven components, five of which are essential for the successful completion of the Planning Coordinator's function as an error recovery unit. These five components are the *Current World Model*, the *Primitive Structure Database*, the *Node/Link Weighting Mechanism*, the *Mapping Mechanism*, and the *Error Recovery Generation Algorithm*. Functioning together, these components provide an architecture suited to the desired function. The architecture itself, however, requires an internal communication and message passing scheme as well as appropriate interfaces to the Intelligent Machine. These two topics are the subject of the next subsection.

3.3 Planning Coordinator Communication and Interface Description

The requirements of the operation of the Planning Coordinator necessitate significant communication between its component parts and between it and the intelligent machine to which it is connected. With respect to the Intelligent Machine Model, the Planning Coordinator must have access to at least the Dispatcher. It is possible for the Planning Coordinator to use the Dispatcher to communicate not only to the Organization Level of the Intelligent Machine but also to the Coordinators and thus, the Execution Level hardware. The communication schemes that need to be implemented are the subjects of the following subsections.

3.3.1 Internal Addressing Scheme

The Planning Coordinator maintains seven components, six of which can be but may not necessarily be physically distinct. It is expected that the Planning Coordinator will migrate to a parallel processing architecture and it is on this basis that a multipoint addressing scheme is desired. As depicted in *Figure 10*, all six of the physically distinct components that comprise the Planning Coordinator maintain communication through a single shared bus. In addition to these six components is a seventh component called an External Communication Link. to be discussed in *Subsection 3.3.3*. A secondary bus exists between the Current World Model, the Node/Link Weighting Mechanism and the Primitive Structure Database, due to the potentially high communication level that will exist between the three components.

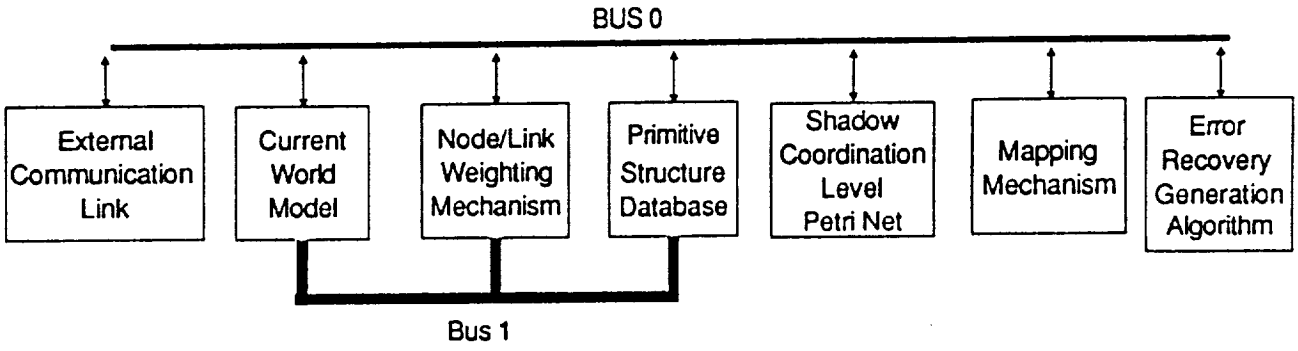


Figure 10
Internal Planning Coordinator Addresing Scheme

The first bus, designated *BUS 0*, is the primary shared bus. Each of the components connected to this bus is assigned a communication address. By definition, this scheme is called a multipoint addressing scheme because more than two physical devices share the same

transmission medium. The duplexity of this communication scheme is multipoint duplex. This indicates that all physical devices are capable of sending and receiving over a single bus.

The internal line discipline of *BUS 0* requires that all devices be considered as peers to facilitate intradevice communication. This discipline is called *contention.* If the communication-line is free, then any device can transmit on it. If the line is not free, then a device must wait. In terms of transmitter-receiver identification, this scheme requires the transmitter to identify both itself and the intended receiver, a technique called *peer multipoint.* The internal addressing scheme presented is aggressively used in computer satellite systems and local area networks. It is often combined with a message passing or packet communication technique to facilitate short duration transmission.

Bus 1 is a dedicated communication bus between the Current World Model, the Node/Link Weighting Mechanism and the Primitive Structure Database. This dedicated bus is necessary due to the potentially high communication demands between the Current World Model, the Node/Link Weighting Mechanism and the Primitive Structure Database. There are two specific times that high communication would load down *Bus 0*, necessitating the existence of *Bus 1*. The first of these times is when the Primitive Structure Database is initially created. The second of these times is when the Current World Model drastically changes, requiring a drastic addition to the Primitive Structure Database. Between these three components of the Planning Coordinator, a transmission scheme that is different from the one used for *Bus 0* is necessary. The internal transmission schemes used by the Planning Coordinator are the subjects of the next subsection.

3.3.2 Internal Transmission Schemes

Transmission via *Bus 0* does not require a relatively continuous flow of data between two specific sources, such as in a voice telephone call or telemetry data communication. Rather the data is transmitted in short duration, high volume bursts among seven different points of communication without the requirement of a dedicated communication link. This type of communication favors the dynamic-packet, message passing transmission scheme which is the transmission choice for *Bus 0* communication. The dynamic-packet, message passing scheme allows the sender and receiver to send dynamic length messages to each other. Length identification, and sender identification are stored in header information contained as part of the message itself. The receiver decodes the sender information, checks the message and acknowledges receipt of the message. Transmission error checking protocol is accomplished through parity checksums.

In contrast to the message passing transmission scheme implemented on *Bus 0*, *Bus 1* utilizes a circuit switching transmission scheme. This is because the communication needed between the components on *Bus 1* are of a long duration, continuous data flow nature. Disruption of the communication can be catastrophic. Again, standard transmission error correction is used, (i.e., hamming codes) and need not be discussed. The two transmission schemes presented are sufficient for the operation of the Planning Coordinator. The next subsection introduces the external communication scheme between the Planning Coordinator and an intelligent machine.

3.3.3 External Communication Link

The External Communication Link is responsible for communication between the Planning Coordinator and the external intelligent machine that the Planning Coordinator is attached to. The External Communication Link, as depicted in *Figure 11*, connects on one side to the Planning Coordinator's *Bus 0*, and on the other side to an external communication port on the Intelligent Machine. Internally the External Communication Link contains a dedicated

processor, a high amount of buffer memory and a dynamically configurable communication port. Functionally the port protocol is established through handshaking between the Planning Coordinator and its intelligent machine counterpart. Electrically it is necessarily expected that both systems will be standard. It is possible to establish an electrical switching protocol in the event that a non-standard configuration is encountered. However, this is considered to be potentially future work and is not considered in this Thesis Proposal.

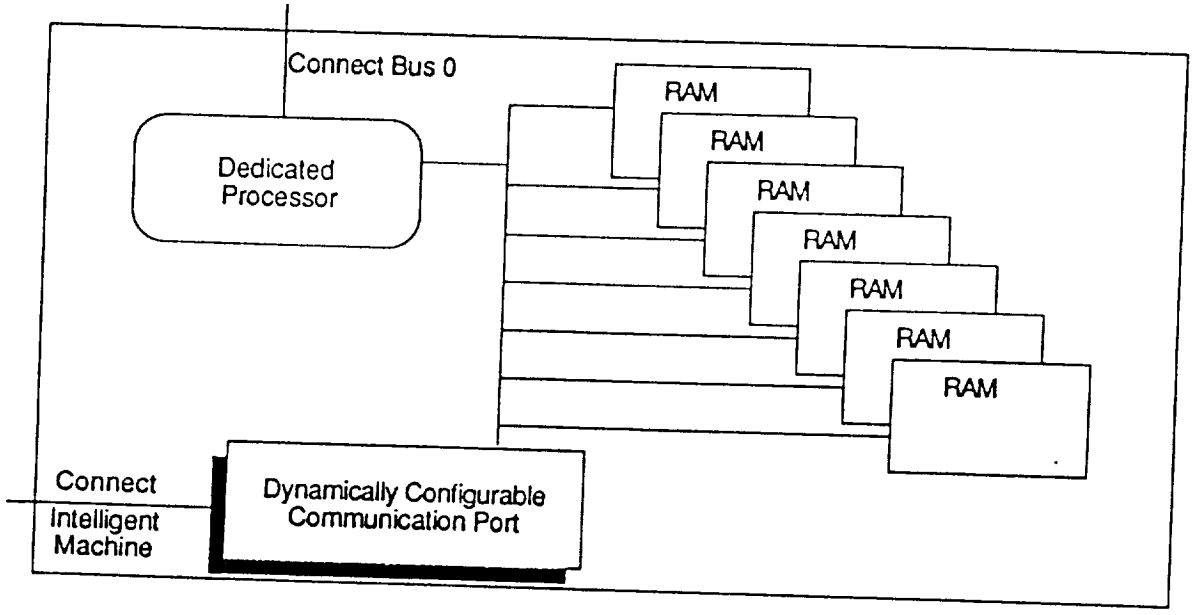


Figure 11
External Communication Link Internal Architecture

3.3.4 Summary

The Planning Coordinator communication and interface description is such that it provides for the growth of the Planning Coordinator through the inclusion of additional components such as the System Fault Monitor. It is robust enough to provide for the communication needs of the Planning Coordinator. Nevertheless there is room for considerable expansion and enhancement of the communication system. The remainder of this section is dedicated to the functional description of the Planning Coordinator, from initial connection to overall operation.

3.4 Functional Description

The primary functions of the Planning Coordinator are to efficiently enact task level error recovery and in the event of catastrophic failure of the main controller of the intelligent machine, to which the Planning Coordinator is attached, act as a minimal backup to the intelligent machine. Note that catastrophic failure of the main controller, is the only non-task level error that the Planning Coordinator is designed to handle and is not to be confused with an irrecoverable error generated by an autonomous task. A secondary function of the Planning Coordinator is to perform on-line planning of autonomous tasks. This function is viewed as an extension to the error recovery function and is treated as such. Functionally, the Planning Coordinator will react differently when presented with differing 'error types.' The differences between error types and the modes of operation that the Planning Coordinator engages in are the subjects of the following subsections.

3.4.1 Error Types: Definitions and Severity

An error, as defined for the purposes of this Thesis Proposal, is an anomaly in the operation and/or execution of a task. The following error types are the general error categories that can be expected during the operation of a task. Note that these error types are defined only in the context of task level operations. They are not defined for the purposes of internal hardware or software glitches. These glitches are the responsibility of the System Fault Monitor and will not be discussed further in this Thesis Proposal.

No Error

No Error is an error type that indicates an on-line plan is needed as opposed to an error recovery. This type of error is identified through the raising of the OP_FLAG flag.

Tolerance Error

A Tolerance Error is an error type resulting from a parameter being out of tolerance by a small margin. The fix for such an error is dependent upon the task being performed. For example, if an insertion task is being performed where the error results from a slight misalignment of the joined parts, the fix for it may be to apply a slight perturbation to the parts until they are back within tolerance. The severity of a tolerance error is low.

Minor Error

A Minor Error is a singularly occurring anomaly resulting from the execution of a task. It is possible that the repeated execution of a task will result in the exact same error recurring. It is for this reason that a frequency counter is associated with an error when the error occurs. In addition, an aging identifier is associated with each error occurrence. This aging identifier is used to distinguish between error occurrences, facilitating a distinction between minor and major errors.

Major Error

A Major Error is defined as a recurring Minor Error, with the number of occurrences exceeding a threshold, within a given time limit.

Irrecoverable Error

An Irrecoverable Error is defined as either:

A single error of either the Minor or Major type that causes a continuous flow of errors, distinct from the original error, but the result of the original error. "Snowballing"

or

A condition caused by the inability of the Planning Coordinator to facilitate an error recovery, or a condition which causes the task operation of the

intelligent machine to forcibly halt, requiring outside intervention. Note that this is different than a catastrophic error.

Upon identification of an irrecoverable error, notification is sent by the main controller of the intelligent machine to a human operator or external information base for assistance.

Catastrophic Error

A catastrophic error is a failure of the main controller of the intelligent machine hierarchy to which the Planning Coordinator is attached. It is identified by the assertion of the MC_FAIL flag. Recovery from this error is accomplished through an attempt to restart the intelligent machine (i.e., via reboot, hard reset, etc.). Failing a restart of the intelligent machine, immediate notification of the problem is sent by the Planning Coordinator to a human operator or other external information base for assistance. In addition, the Planning Coordinator attempts to place the intelligent machine hierarchy in a quiescent state until assistance arrives.

These error types are used by the Planning Coordinator to perform its error recovery. The following subsection examines the error recovery mode of operation of the Planning Coordinator.

3.4.2 Error Recovery

Error recovery is the default mode of operation of the Planning Coordinator. While in this mode, the Planning Coordinator is in either an *observer* or an *actor* state. The Planning Coordinator's default state is that of an *observer*, monitoring the execution of the Intelligent Machine's Coordination Level Petri Net through the previously described Shadow Coordination Level Petri Net. While in this state, the Planning Coordinator is a strict monitor and does not interfere in the operation of the Intelligent Machine. The Planning Coordinator changes to the

actor state only when an error occurs. Its active role is limited to the execution of task level error recovery plans.

The operation of the Planning Coordinator while in error recovery mode is outlined in the block diagram descriptions of *Figure 12*. Each of the four main blocks is described below.

Block B1

The Planning Coordinator periodically monitors the action of the Shadow Coordination Level Petri Net, waiting for an interrupt indicating the assertion of ERR_FLAG, OP_FLAG, or MC_FAIL. While none of the flags are asserted, the Planning Coordinator engages in other functions. These other functions include update of the Primitive Structure Database, and creation of most likely errored events.

Block B2

Once an error recovery interrupt has occurred, the Planning Coordinator status is changed from *observer* to *actor*, and the Planning Coordinator begins its error recovery function. First the Planning Coordinator attempts to qualify the error by obtaining error information from the intelligent machine. If the information is available, the Planning Coordinator reacts to it and creates the appropriate error recovery route utilizing the appropriate section of the Error Recovery Generation Algorithm.

Note 1: It is because the Planning Coordinator utilizes the Shadow Coordination Level Petri Net and the Mapping Mechanism to identify the location of the error that an error recovery can be enacted quickly. This structure allows for the [complete] minimization of the search space that the Planning Coordinator would otherwise have to search to characterize the error and determine a course of action to take.

Note 2: The Planning Coordinator is necessarily subservient to the main controller of the intelligent machine to which it is connected. This limitation permits the main controller to actively abort any error recovery.

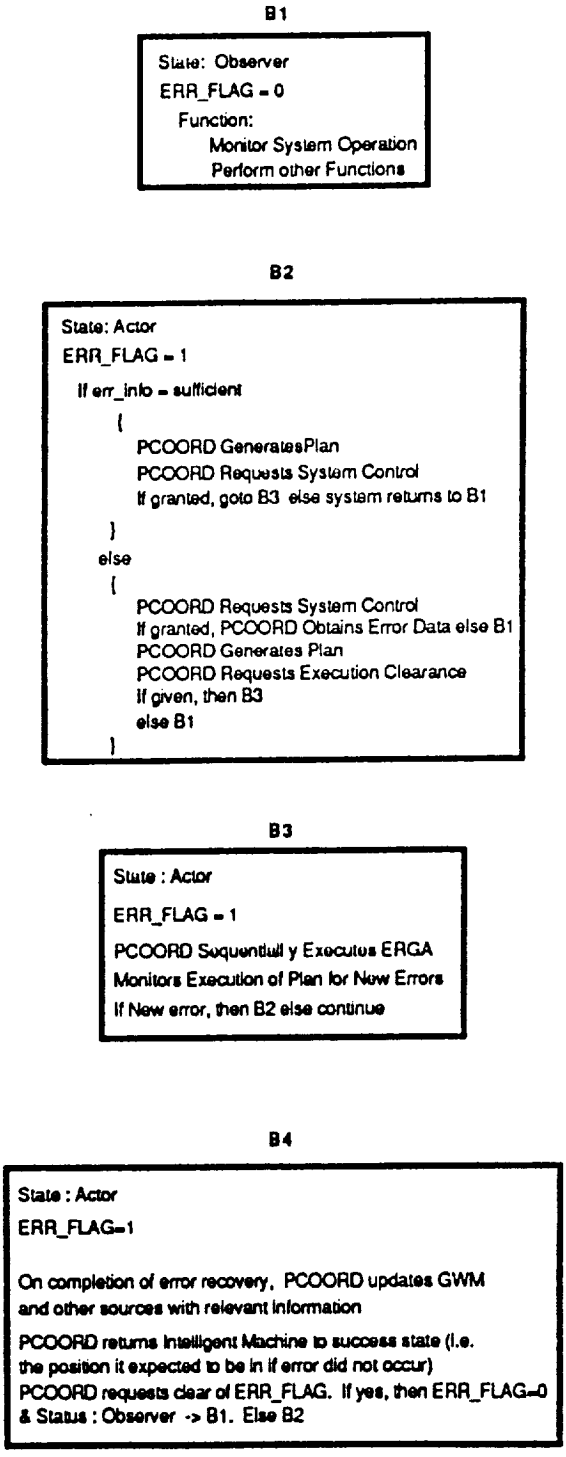


Figure 12
Operation of Planning Coordinator in Error Recovery Mode

Block B3

The Error Recovery Generation Algorithm has been used to create an error recovery plan. Each component Primitive Structure is monitored for error as it executes. If a new error is caused during the error recovery, it is handled in the same manner as any other error with the exception that the new error is handled first. Requests for system control are again made to the main controller, but in the case of nested errors, the main controller is the Planning Coordinator.

Block B4

Once an error recovery has been completed, it is the responsibility of the Planning Coordinator to update all relevant information to correctly reflect the state of the intelligent machine prior to returning control of the intelligent machine to the main controller. Once control has reverted back to the main controller, the Planning Coordinator's status reverts back to *observer*.

This concludes the functional description of the Planning Coordinator during its error recovery mode. A second mode of operation that the Planning Coordinator engages in is on-line planning. This Thesis Proposal advocates the structuring of the on-line planning as a specific instantiation of an error recovery. On-line planning is the subject of the next subsection.

3.4.3 On-Line Planning

The need for on-line planning in an autonomous intelligent machine is indisputable. The mechanisms for realizing on-line planning are many. The Planning Coordinator architecture recognizes on-line planning as a specific instantiation of an error recovery. This is due to the fact that an on-line plan must use the same basic information required by an error recovery plan (i.e., the Current World Model and Primitive Structure Database). However, the active information needed by an on-line plan is not as varied as that needed by an error recovery plan. Often the information needed is a verification of, or modification of, specific parametric

information. With respect to the Planning Coordinator, on-line planning is broken up into two sub-categories: *Short Term On-line Planning* and *Interactive On-line Planning*.

3.4.3.1 Short Term On-line Planning

Short Term on-line planning is performed while the Planning Coordinator is in an *observer* state. It results from an insufficiency in the information available when the original long term, off-line plan was created by the Planning Level, here the Organization Level, of the intelligent machine architecture. In this case, the needed information is available after some sequence of events has taken place. Effectively, the on-line planning takes advantage of the Planning Coordinator's error recovery platform to create on-line plans dynamically. This procedure is initiated by the sole assertion of the OP_FLAG.

The Organizer must recognize the need for an on-line plan. Thus within the structure of the Petri Net passed down by the Dispatcher, a transition is defined to be an on-line plan transition. When the Shadow Coordination Level Petri Net is created, this transition is expanded such that when it becomes active, the OP_FLAG is asserted. Thus using the existing parameters and the Error Recovery Generation Algorithm, an on-line plan is created and executed. In this case, the on-line plan will likely correspond to the most likely errored event and can be preprocessed. If an on-line plan is in the process of being pregenerated and an error recovery is needed, the error recovery will take precedence. Interactive on-line planning is handled similarly and is the subject of the next subsection.

3.4.3.2 Interactive On-line Planning

The need for Interactive On-line Planning presumes that expected conditions can change from the time an off line planner creates its plan, to the time the plan is actually executed.

The changes are often in the parameters under which the task must operate. Changes in these parameters may or may not induce an error. The original off line plan must be capable of realizing that particular parameters may change, and as such allow for the verification of those parameters.

In this case, the off-line planner permits a transition to be created, such that the transition simultaneously activates the ERR_FLAG and the OP_FLAG. When the Planning Coordinator sees this, it understands that a parameter verification is required and executes a specialized function which takes as input the parameters to be verified, and provides as output the corrected parameters. It is the function of the main controller to determine whether these parameters are valid or not. If they are not, then an error recovery may be initiated. If they are, then the system continues operation unimpeded.

The operation of the Planning Coordinator, from connection to the Intelligent Machine through error recoveries and reconfiguration of the Current World Model and Primitive Structure Database is demonstrated in the comprehensive example of the following subsection.

3.5 Example Operation of the Planning Coordinator

The following subsections present examples of the operation of the Planning Coordinator. In each case, the intelligent machine under consideration is explained as fully as possible. In addition, the components of the Intelligent Machine Model are listed for clarification.

3.5.1 Robotic Assembly Workcell: Strut Insertion

The following example uses the dual arm, vision aided, robotic testbed in the Center For Intelligent Robotic Systems For Space Exploration (CIRSSE) as its basis. As such, the Planning Coordinator is used as a supplement to the Intelligent Machine Model upon which the CIRSSE testbed is based. The testbed was initially designed for application of space related robotic functions but also has application to terrestrial robotics. It features two, six degree of freedom robotic arms, each secured to a pan-tilt platform that is itself mounted to a linear-track cart. This provides nine degrees of freedom for each of the robotic arm assemblies. Mounted on one of the robotic arms are two cameras to assist in the positioning of the arm for grasping, and to assist in general stereo vision applications. Mounted on the ceiling are two all purpose cameras and a laser scanner, used for general position identification. Force sensors provide feedback from each non-articulated gripper. Computer control is provided by an Etherlinked Sun station computer network used for code generation and compilation. A Motorola VME cage is used to execute the object modules, representing the operational code which are dynamically linked when loaded. *Figure 13* indicates the relative positions of each of the testbed components, a table that is used to hold structures, and a strut holder used to hold struts that are later taken as finished material for insertion.

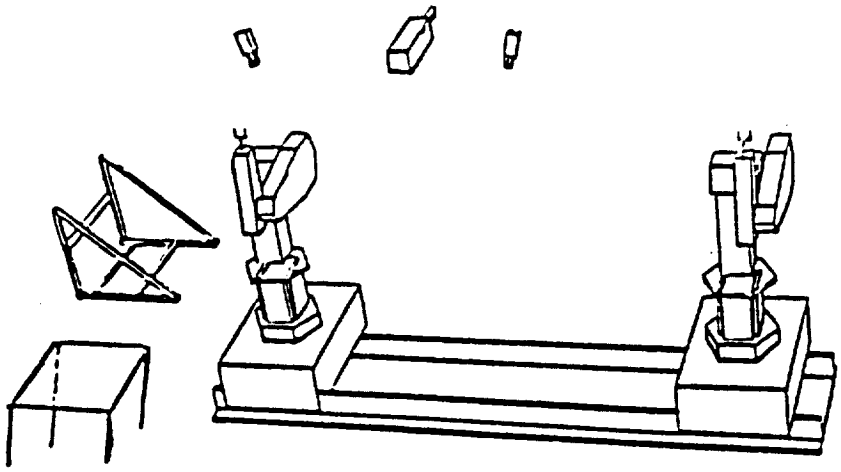


Figure 13
CIRSSE Testbed

Figure 14 presents a comprehensive block diagram of the Intelligent Machine Model representative of the CIRSSE testbed. This diagram logically includes the Planning Coordinator. With respect to the overall CIRSSE testbed, the Planning Coordinator connects to the VME cage through a communication port on one of the communication cards inserted into the VME cage. It is assumed for this example that this is the first time the Planning Coordinator is connected to the CIRSSE testbed.

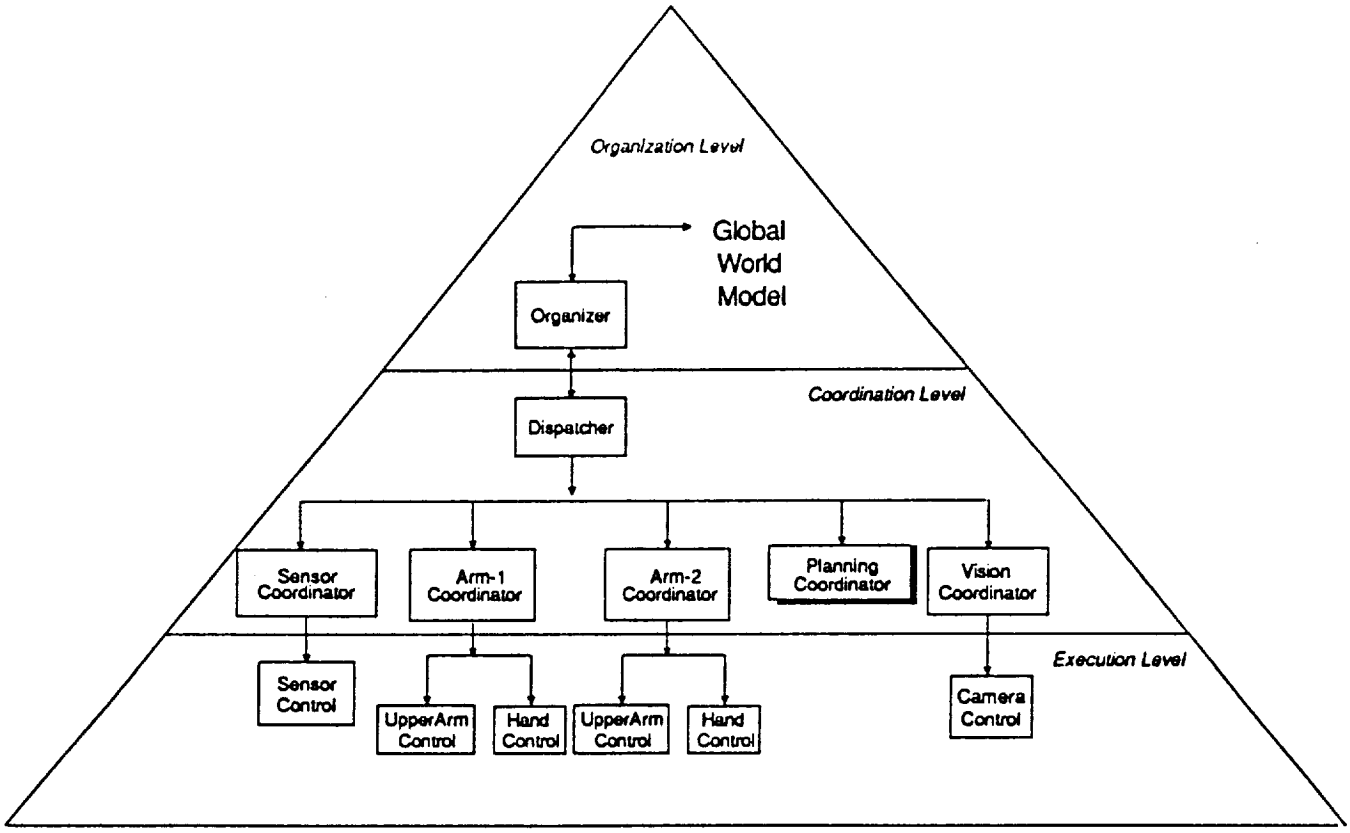


Figure 14
The Intelligent Machine Model Hierarchy Of CIRSSE Testbed. Includes PCOORD

The first step is to establish communication between the CIRSSE testbed and the Planning Coordinator. This is accomplished by first physically connecting the Planning Coordinator to

the CIRSSE VME cage. Once this has been accomplished, the connection causes an I/O port interrupt. This results in the initiation of handshaking protocol between the Planning Coordinator and the CIRSSE testbed main controller, through an interrupt service routine, a function of the CIRSSE testbed. The main controller sends out a query to the testbed communication port at which the Planning Coordinator resides. The Planning Coordinator interprets the query and dynamically configures its communication port to match the parameters of the query. It then responds to the main controller that it is active and negotiates the frequency of a *keep alive* communication signal. This signal is passed between them at regular intervals, sufficient to identify the liveness of the components, but not enough to bog down either the Planning Coordinator or the CIRSSE testbed communication schemes. This structure is used by the Planning Coordinator to ascertain if the MC_FAIL flag should be asserted. If multiple *keep alive* signals sent from the Planning Coordinator are not responded to by the CIRSSE testbed, the CIRSSE testbed is considered to have failed.

In addition to the *keep alive* signal, the Planning Coordinator requests two other major pieces of information from the CIRSSE testbed main controller. The first piece of information is the error identification code transmitted when an error occurs during task operation. *Note that it is assumed here that such information is available and will be transmitted along the main communication bus between the Dispatcher and Coordinators of the Coordination Level.* The Planning Coordinator, through its External Communication Link, monitors the communication bus and uses this information to assert ERR_FLAG, identifying the start point for its error recovery scheme. The second piece of information requested is a dump of the GSCPN created by the Dispatcher when the Dispatcher creates an Operational Petri Net for task execution. This dump facilitates the creation of the Shadow Coordination Level Petri Net. After receipt of the error code and acknowledgment of the request for GSCPN, the Planning Coordinator requests a dump of the Global World Model from which the Current World Model is constructed. Since it

is the first time that the Planning Coordinator has been made active, the information stored in the Current World Model is identical to that stored in the Global World Model.

The Planning Coordinator stores the transmitted data from the Global World Model. It is assumed that the Global World Model is a geometric representation of the objects that exist in the environment of the intelligent machine. In addition to the geometric representation, it is expected that the Global World Model will have some idea as to the functions performed by the known objects in the environment. Under these assumptions, the Global World Model is transformed into a linguistic representation of objects and functions, where the linguistic representation implements the *case_frame* construct as the representation tool. This is the Current World Model. Upon completion of the transformation of the Global World Model information, the Planning Coordinator begins to derive the underlying semantic network for the Primitive Structure Database, and the Dynamic Fuzzy Rule Base. The Dynamic Fuzzy Rule base begins with rules generated from the basic structure of the intelligent machine under consideration, akin to the instincts that humans possess from birth. As such, there is always a predetermined starting point for the Rule Base. Using either an expert system or specified combining techniques these rules are augmented. In parallel with the development of the Fuzzy Rule Base, the underlying semantic network representing the Primitive Structure Database must be derived. This is done by *reading* the Current World Model and interpreting objects and events from the data stored there. *Note that there is an inherent difficulty here. If the Global World Model does not have an informed geometric representation of the environment (i.e., an understanding of the basic objects in the environment and how they function), then it will not be possible to create a sufficient linguistic representation of the information. This will prevent the autonomous creation of the underlying semantic network for the Primitive Structure Database, and it will not be possible to derive the Fuzzy Rule Base. Here it is assumed that the Current World Model possesses the desired qualities.*

From the Current World Model's linguistic representation of objects and functions, the object and event nodes of the underlying semantic network are created. In particular, it is from the geometric information of the Global World Model that the object nodes are created and it is from the knowledge of their function(s) that event nodes are formed. These nodes are qualitatively linked together to form an unweighted semantic network through the linguistic parsing of the representations in the Current World Model. Each node still represents a *case_frame*. However, from the parsing, the *case_frames* are identifiable as either object or event nodes. For each of the nodes created for the semantic network, a mirror Primitive Structure Interface Node is created for the Mapping Mechanism.

When both the underlying semantic network and the dynamic fuzzy rule base are completed, construction of the Primitive Structure Database begins. Starting from any node in the semantic network, label the node and follow one of the directed links to a connected node. Take both nodes and the connecting link and apply them to the Fuzzy Rule Base. If both nodes are object nodes, then the nodes represent a start point and an end point for the link between. After applying the nodes and link to the Fuzzy Rule Base, a fuzzy weight, representing the strength of the connection is determined and applied to the connecting link. If one node is an object node and one node is an event node, then a search from the event node is initiated to find all connected object nodes. The triple formed by the original object node, the event node, and individually, each of the other object node(s), along with their two interconnecting links are then applied to the Fuzzy Rule Base. This results in a fuzzy weight applied to the connecting links. This procedure is repeated until all nodes are labeled and all links are weighted. The Primitive Structure Database is now complete. No specific labeling algorithm has been chosen. A minimal time one is suggested. *Figure 15* and *Figure 16* depict respectively, a representative but not comprehensively expanded Global World Model, and a Current World Model. It must be noted that the functional specification of the object and event nodes is contained in the *case_frame* represented by the nodes. It is not explicitly shown in the linguistic model of the

Current World model, except in the identification of a word as an object node or event node, as per *Figure 16*. *Figures 17* through *20* depict a representative underlying semantic network and the result of applying the underlying semantic network nodes and links to the Fuzzy Rule Base, the Primitive Structure Database.

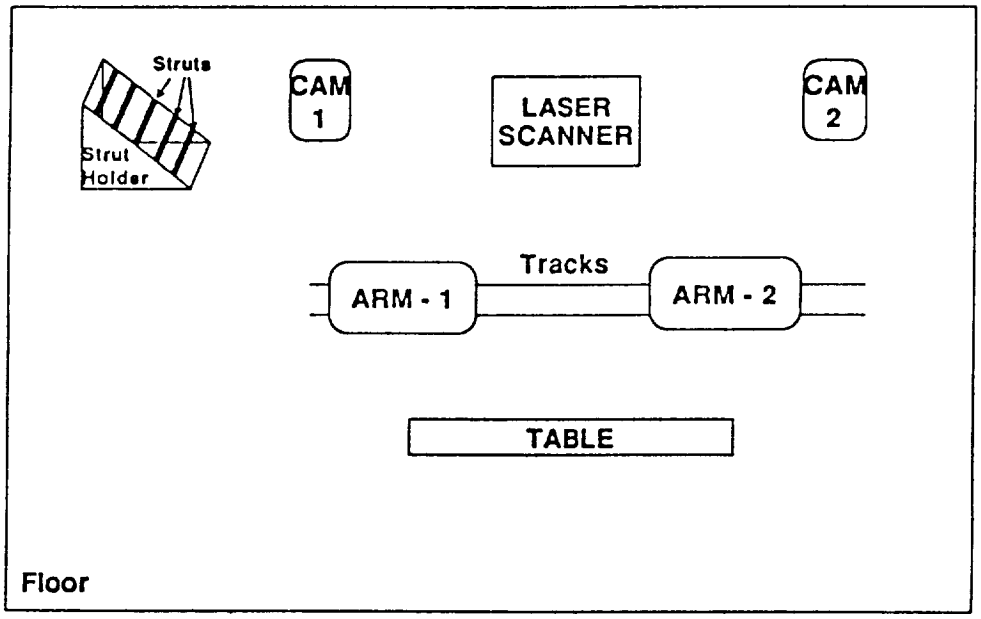


Figure 15
Representative Global World Model

_Strut _Holder _seated_ on _floor.
_Cam_1 _suspended_ from _ceiling.
_Cam_2 suspended from ceiling.
_LASER_SCAN suspended from ceiling.
_Arm_1 _mounted_ to tilt _bracket on linear _track.
_Arm_2 mounted on tilt bracket on linear track.
Linear track mounted to _floor.

_Word = Object Node -> Object Case_Frame
Word = Event Node -> Event Case_Frame

Figure 16
Representative Current World Model

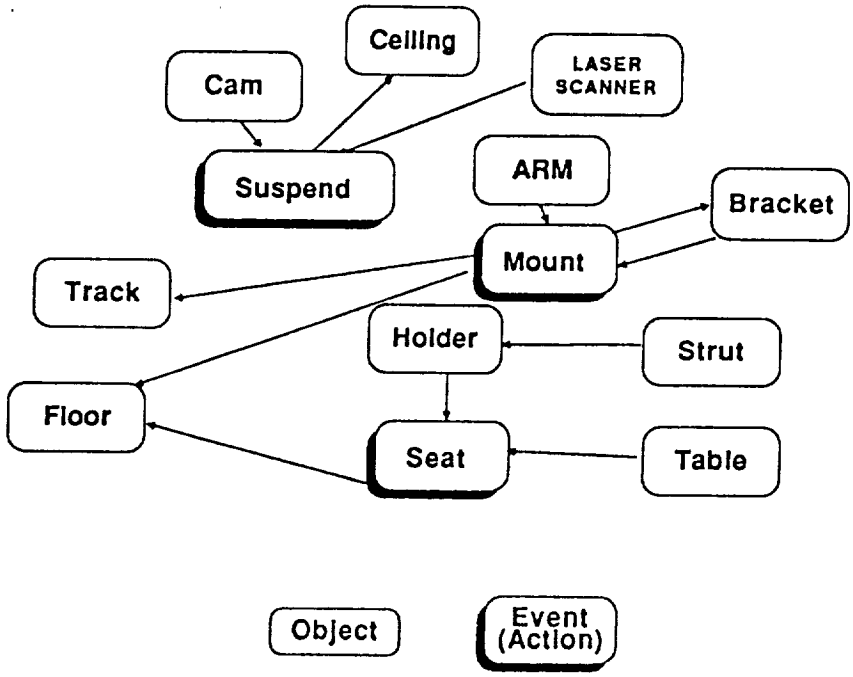


Figure 17
Representative Underlying Semantic Network With Unweighted Links

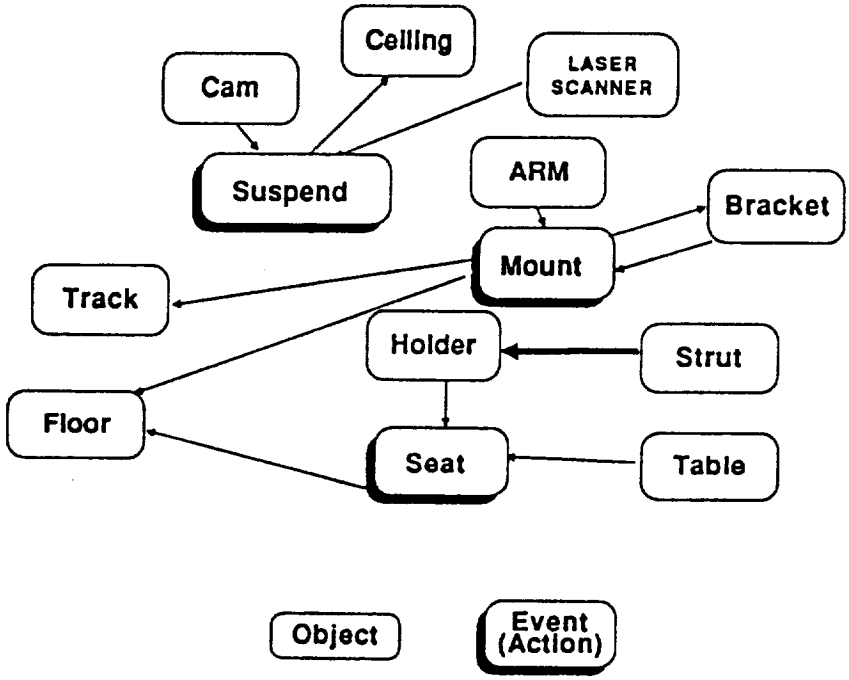


Figure 18
Representative Weighting of Object - Object Connection Depicted With Bold Line

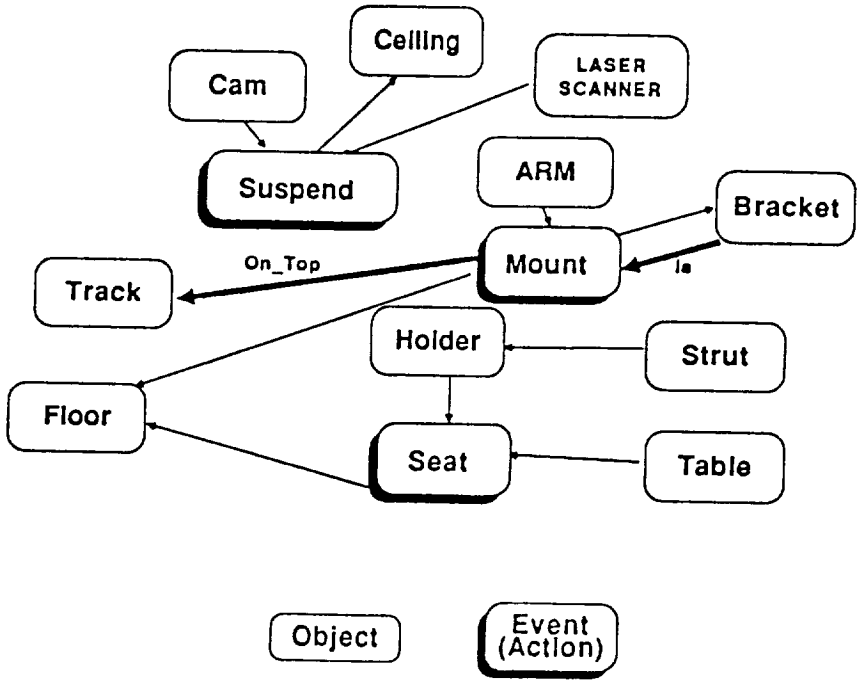


Figure 19
Representative Linguistic Weighting of Object - Event - Object Connection Depicted With Bold Lines

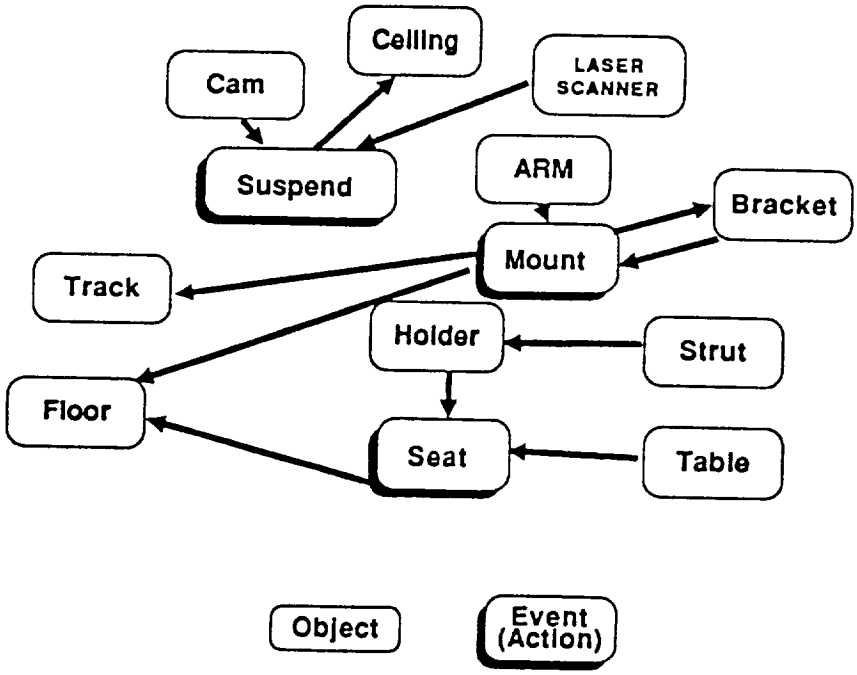
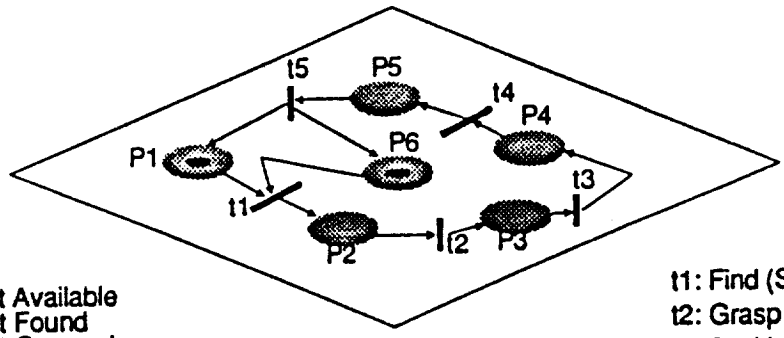


Figure 20
Representative Primitive Structure Database
Note: For Diagram Clarity, linguistic weighting not shown

This work has been done prior to the actual beginning of the operation of the intelligent machine. At this point, the intelligent machine begins operation. Its first task is to create a triangle by inserting a single strut into a *v-shaped* strut construct located on the table in the assembly workcell. A GSCPN representing this operation is generated by the Dispatcher and is depicted in *Figure 21*. Due to the previous code identification setup, an equivalent Shadow Coordination Level Petri Net is generated from the GSCPN. The transitions of the SCPN are immediately connected to their respective Map Interface Nodes. The SCPN is depicted in *Figure 22*, along with the Map Interface and Primitive Structure Interface Nodes corresponding to the SCPN transitions and primitive structures, respectively. Also during this time, the most likely errored event recovery net is generated and is depicted in *Figure 23*.



P1: Strut Available
P2: Strut Found
P3: Strut Grasped
P4: Strut Positioned
P5: Strut Inserted
P6: Arm Available

t1: Find (Strut)
t2: Grasp (Strut)
t3: Position(Strut)
t4: Insert(Strut)
t5: Verify(Insertion)

Figure 21
GSCPN Generated For Strut Insertion

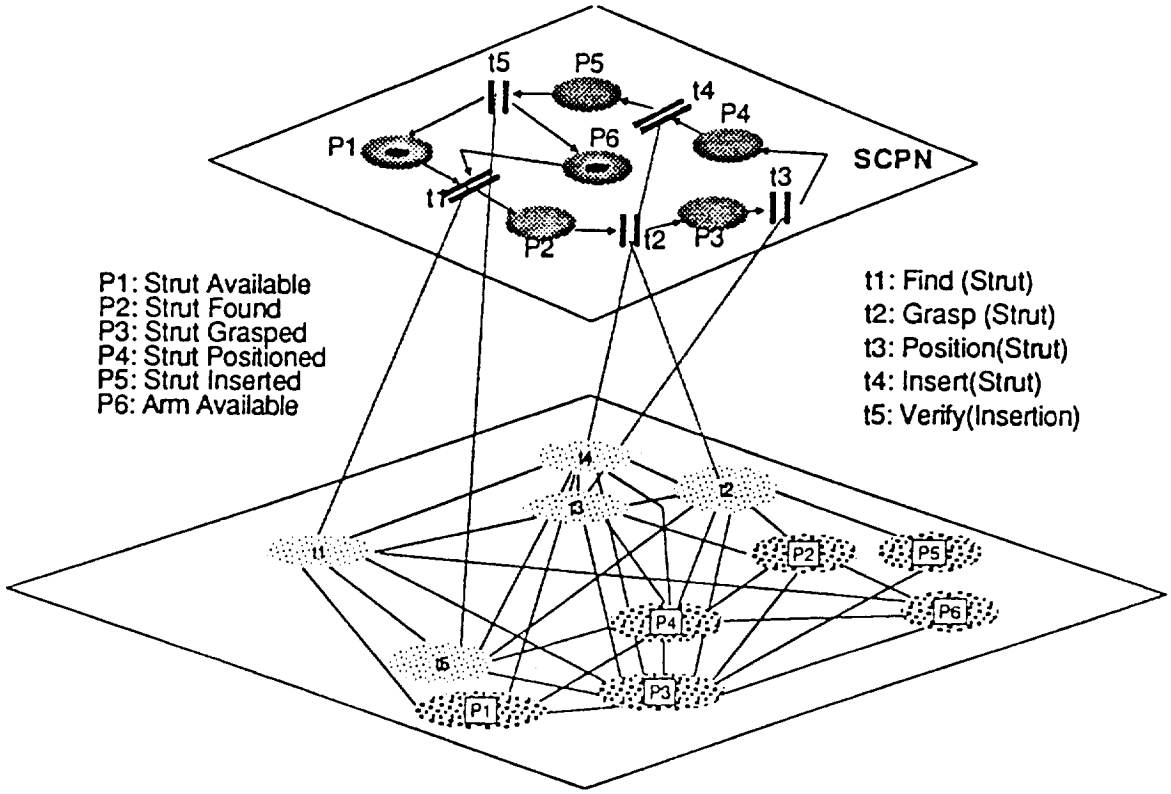


Figure 22
SCPN Generated For Strut Insertion

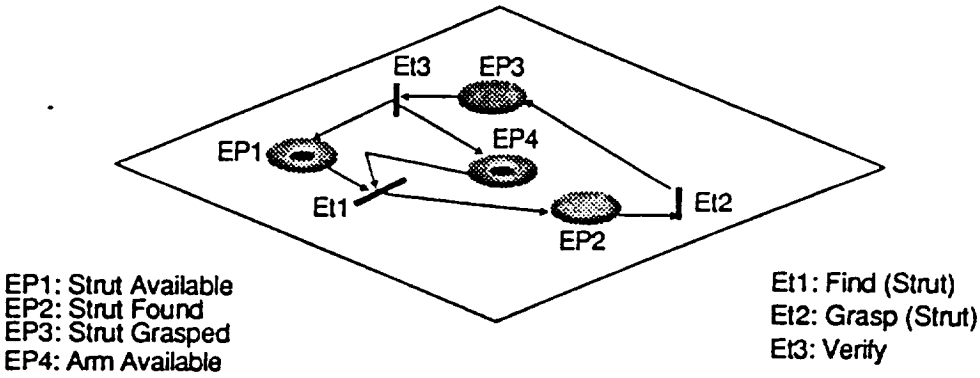


Figure 23
Most Likely Errored Event Recovery Net
For Transition t2 of SCPN

This is accomplished by using the nodes on either side of the SCPN transitions as the start and end nodes of a recovery, identifying their equivalent nodes in the PSDB and beginning the search procedure previously outlined. At this point, the intelligent machine begins functioning. From *Figure 21* and *Figure 22*, it is apparent that there are multiple points at which errors can occur, since an error can occur at any transition in the Petri Net. For the purposes of this example, the following three errors are considered possible:

1. There is no strut available in the strut holder.

2. The strut is grasped incorrectly and along the way to inserting it, the strut is dropped.

3. The strut is inserted, but is inserted incorrectly, requiring a slight perturbation (tolerance error) to put it into place.

Consider the first error. The intelligent machine attempts to get a strut. One of the Global World Model objects is a strut holder from which struts are dispensed. Since the only other struts the intelligent machine knows about are connected in a *v-shape* on the table, the only strut source is the strut holder. Error number one occurs. This type of error is initially a Minor Error, as per the definition. The intelligent machine sends out an error code indicating an error has occurred. The External Communication Link of the Planning Coordinator picks up the error code and raises ERR_FLAG. The Planning Coordinator notifies the main controller of the intelligent machine that it wishes to perform an error recovery. *Note that although the Planning Coordinator is subservient to the main controller in the hierarchy of the intelligent machine the Planning Coordinator will still notify the intelligent machine main controller that it wishes to perform an error recovery. This is to ensure that the main controller has received the error code generated by the intelligent machine. By design, the Planning Coordinator will not perform the error recovery until the request is granted. If the request is not acknowledged, the Planning*

Coordinator will request again. This will continue until the request is acknowledged in the affirmative or in the negative. Here, the request is granted. The error recovery begins.

Initially, the intelligent machine wanted to get a strut. This function is complex and can be decomposed into either a hierarchical, or a sequential series of more primitive actions. Such would have been done in the generation of the CWM and PSDB yielding the GSCPN in *Figure 24*. Hence when the error recovery begins, the first node of the error recovery sequence would be the *find, case_frame.*

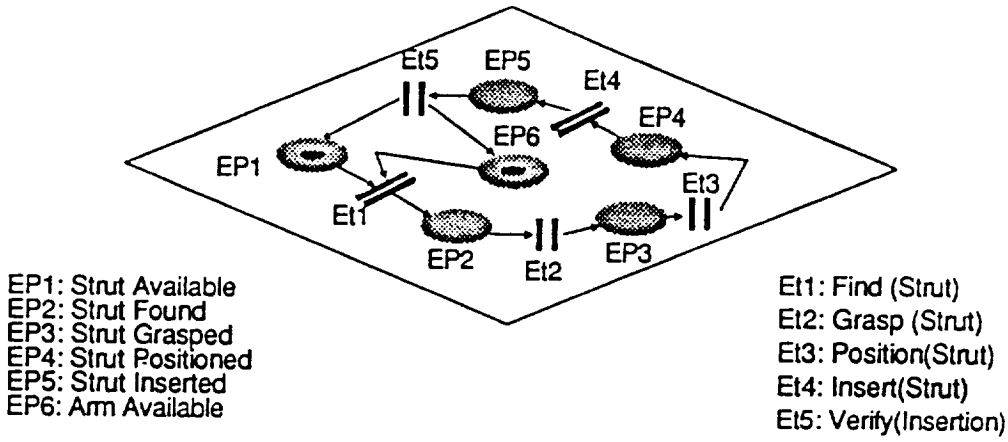


Figure 24
SCPN Error Recovery Net Generated For Strut Insertion

Findable objects would be one of the internal objects of the *case_frame.* There are two possibilities; either there are findable struts or there are none. Assume first that there are no

findable struts. Thus, another error occurs. This error is a nested one and will result in another error and another, all of which are the result of the initial error. After the first several errors, within a given time frame, the Minor Error is reclassified as a Major Error. The errors continue, becoming a cascade of nested errors. The Planning Coordinator recategorizes the Major Error as an Irrecoverable Error and notifies the main controller of the intelligent machine. The intelligent machine takes this information and must request assistance to obtain a solution. Assume next that there are findable struts. The recovery net represented in *Figure 23* is executed, yielding at its conclusion a strut in the grasp of the intelligent machine. First the strut is found, its location being obtained and passed to the *case_frame* represented by *move_to*. The next state is that the strut is grasped. This is accomplished and the strut is ready to be inserted.

Consider now that error number two occurs. The strut is grasped incorrectly and along the way to inserting it, the strut is dropped. This condition may or may not be noticed until the strut is to be inserted. Either way, the solution is the same. As before a Minor Error occurs. The last good state in the overall operation occurred when the strut was grasped. This is part of the precondition that the strut was grasped. Hence, this function is performed again. First a strut is found. In this case, the strut location is different than before. Assume that an attempt to grasp the strut is made but cannot be done. A second Minor Error has occurred, this due to the fact that the robot arm could not reach the found strut. This nested error is handled first. A new strut is found and is obtainable. This strut is obtained. This not only solves the nested error recovery, it solves a part of the initial error recovery. Once in the gripper, the strut is moved to the insertion point. *Figure 25* shows the sequence of actions outlining this recovery. Note, the number of consecutive errors that would force a Minor Error into a Major Error is variable. Here, two consecutive errors is not sufficient to cause a Major Error. It should also be noted that unless the struts have some kind of identifying number or marking, there is no way that the intelligent machine can determine which strut it has picked up. As a result there may be no garbage collection unless specifically requested (i.e. in a new plan).

C-2

Finally, consider that error number three occurs. The insertion of the strut appears to have gone according to plan. However, on verification of the strut position by the intelligent machine, it is found that the strut is not seated completely in its connector. The intelligent machine sends out an error code. The error code is picked up by the External Communication Link of the Planning Coordinator. The error information indicates that the error type is a tolerance error. This type

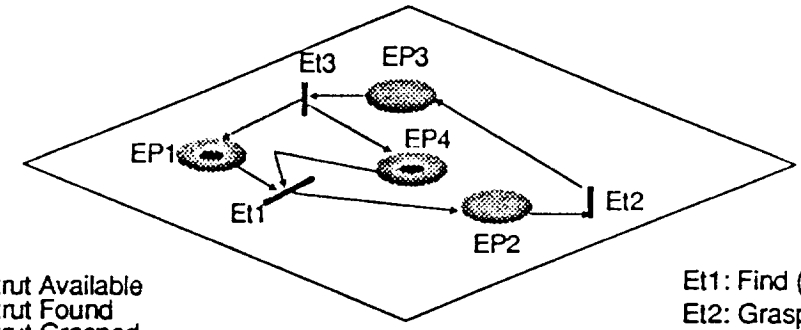


Figure 25
Error Recovery Net For Error Number 2

of error has a predefined recovery plan based on the last activity performed by the intelligent machine. The recovery plan looks at what the results of the previous task are supposed to be (i.e., the postconditions) and examines each one until a violation is found. Here the violation is, *strut insertion not locked in connector*. Based on this violation the strut is further pressed until it locks in place. The locking is verified by a verification process.

Once the error recovery has been performed, the intelligent machine returns to operation as though there were no break in the continuity of the operation. This concludes the example. The following section outlines the research goals and proposed work for the remainder of this thesis.

4.0 Research Goals and Proposed Work

The goal of this research is to provide a comprehensive, stand-alone, design architecture for the establishment of robust, task level, error recovery and on-line planning on a single, integrated platform. Although there has been previous work in error recovery and on-line planning, there has been none that incorporates the two on a single platform, or defines explicitly at what level in an intelligent hierarchy the two should be logically positioned.

4.1 Contributions To Date

The following contributions toward realizing the goal of this thesis have been made:

- The design of the Planning Coordinator has been provided, which incorporates task level error recovery and on-line planning on a single integrated platform [Farah 92a].

- The level in an intelligent machine hierarchy at which error recovery and on-line planning should logically take place has been identified. Previous work had placed error recovery at too low a level in the operating scheme of an intelligent machine hierarchy [Farah 92a].

- On-line planning as a specific instantiation of error recovery has been proposed and is facilitated by the design of the Planning Coordinator [Farah 92a].

- The capability of either telepresent operation or autonomous operation of the Planning Coordinator functionality is designed into the Planning Coordinator architecture, through the Planning Coordinator's Error Recovery Generation Algorithm [Farah 92a].

- The application of Semantic Networks for task level error recovery and on-line planning of intelligent tasks has been introduced [Farah 92b].

- A non-combinatorially explosive means of storing relatable information for planning has been introduced [Farah 92b].

- A means of identifying robust plans within a Semantic Network and retrieving them has been introduced [Farah 92b].

- The use of Fuzzy Logic Techniques in combination with Semantic Networks to create a Fuzzy Semantic Network used for error recovery and on-line planning has been introduced [Farah 92c] .

- The incorporation of both the links between object and/or activity nodes of a Semantic Network, and the nodes themselves to create a fuzzy weighting value is introduced, providing for the possibility of numerous different connections between any two nodes [Farah 92c].

- An algorithm for the creation of a hierarchically organized plan execution list in which an overall possibility of success number is

used to determine plan ordering for execution during an error recovery is introduced [Farah 92c], and refined [Farah 92d].

- The type limitation on the semantic network has been established for the Primitive Structure Database. This limitation forces the semantic network to deal with non-abstract concepts and actions.

- An organism is provided from birth with innate capabilities and knowledge about itself, its functions and its capabilities, regardless of the environment in which it must function. Based upon this core of knowledge, the organism is capable of deriving new functionality. Similarly, an intelligent task level error recovery system must start with a core of primitive structures defining itself, its functions and its capabilities, and a core of rules. Together, these cores form the knowledge base and restrictions from which new functionality is derivable.

4.2 Expected Contributions

Investigation of the error recovery problem has revealed several challenging research areas: *the overall design architecture of the Planning Coordinator, the creation and maintenance of the Primitive Structure Database, the identification and retrieval of plans from the Primitive Structure Database, and the creation and maintenance of the Dynamic Fuzzy Rule Base.* The contributions listed in the previous section are the starting point for a more comprehensive treatment of each of the areas. Anticipated contributions of this thesis include the following.

- Completion of the design architecture that is called the Planning Coordinator through the identification of any further components required for error recovery and on-line planning.

- Introduction of new, and refinement of previously introduced, algorithms and procedures for:
  1. Error Recovery Generation Algorithm
  2. Plan Execution List Generation Algorithm
  3. Primitive Structure Database Generation Algorithm

  etc.

- Identification of the limitations of the Planning Coordinator Design Architecture, with a focus on the Primitive Structure Database construct.

4.3 Proposed Work for the Thesis

There are four major research areas that this thesis may approach. They are:

1. The overall design architecture of the Planning Coordinator.
2. The creation, maintenance of, and plan identification and plan retrieval from the Primitive Structure Database.
3. The creation and maintenance of the Dynamic Fuzzy Rule Base.
4. The establishment of the Global World Model to Current World Model transformation.

The following addresses the first two of these four areas, and describes the anticipated work to be done.

- **Complete the Parallel Planning Coordinator Design Architecture**: The Planning Coordinator is intended to be a parallel design architecture. The completion of the parallel design architecture necessitates:
  1. Identification of the parallelizable components of the Planning Coordinator.
  2. Identification of additionally required handlers (i.e., components needed to administer the overhead of the parallel architecture) for example: Bus-Master, External Communication Link, etc.
  3. Refinement of inter-component communication design based on perceived communication needs.

4. Refinement of the existing Planning Coordinator components in order to facilitate parallelizability, where needed.

- **Establish the Limitations of the Planning Coordinator's Primitive Structure Database:** The focal point of this thesis work is the Primitive Structure Database. While there are several components and interfaces within the Planning Coordinator that are in and of themselves research areas, it is the opinion of this student that the crux of the Planning Coordinator design is the Primitive Structure Database.

  To establish the limitations of the Primitive Structure Database, it is necessary to analyze the Primitive Structure Database with respect to:

  1. Memory Requirements
  2. Size Complexity
  3. Search Time Complexity
  4. Node Constraints: Types of nodes that can be accommodated by this design (i.e. abstract concepts versus physical concepts -> God versus a gripper).

- **Evaluate the Planning Coordinator Architecture:** Although no other architectures which incorporate error recovery and on-line planning into a single integrated platform have been uncovered, it is necessary to measure the Planning Coordinator architecture against some standard. This standard will necessarily be

intelligent machine architectures and architectures which claim to provide either error recovery or on-line planning.

The criteria for comparison will be based on :

Component choice:

- Component type
- Component complexity
- Number and complexity of levels.

Tradeoffs:

- Design criteria (i.e., remote versus non-remote operation)
- Design flexibility
- Speed versus added robustness
- Coherence of differing component levels
- Memory requirements

References

[Akatsu 91] Akatsu, M. , Murata, T., and Kurihara, K., "Verification of Error Recovery Specification for Distributed Data by Using Colored Petri Nets," IEICE Transactions, E 74, 10, pp. 3159 - 3167, October 1991.

[Albus 75] Albus, J.S., " A new approach to manipulation control: the cerebellar model articulation controller," Transactions of the ASME, Journal of Dynamics Systems, Measurement and Control, 97, pp. 220-227.

[Albus 83] Albus, J.S., Mc Lean, C.E., Barbera, A.J. and Fitzgerald, M.L., " Hierarchical Control for Robots in an Automated Factory," Proceedings of the Thirteenth ISIR Robots 7, 2, 13:29-43, April 1983.

[Albus 90a] Albus, J.S., "A Theory of Intelligent Systems," Proceedings of the 1990 IEEE Conference on Intelligent Control, Philadelphia, Pennsylvania, September 1990.

[Albus 90b] Albus, J.S., "The Role of World Modeling and Value Judgment in Perception," Internal Document, National Institute of Standards and Technology, May 1990.

[Brachman 85] Brachman, R.J., "On the epistemological status of semantic networks," Associative Networks, (N.V. Findler,

Editor),Academic Press, Florida (1975). Reprinted (1985, Readings in Knowledge Representations, Morgan-Kaufman, California.

[Brooks 82] Brooks, R.A., "Symbolic Error Analysis and Robot Planning," International Journal of Robotics Research 1,4, 1982.

[Brooks 86] Brooks, R.A.," A Robust Layered Control System for a Mobile Robot," IEEE Journal of Robotics and Automation 2, pp. 14-23, April 1986.

[Cao 91] Cao, T. and Sanderson, A.C., "Task Sequence Planning in a Robot Workcell Using AND/OR Nets," Proceedings of the IEEE International Symposium on Intelligent Control, pp. 239-244, Arlington, Virginia, August 1991.

[Cao 92] Cao,T. and Sanderson, A.C.," Sensor-based Error Recovery for Robotic Task Sequences Using Fuzzy Petri Nets," Proceedings of the 1992 IEEE Robotics and Automation Conference, Nice, France, May 1992.

[Chen 90] Chen, S., Ke, J. and Chang, J., "Knowledge Representation Using Fuzzy Petri Nets," IEEE Transactions on Knowledge and Data Engineering 2, 3, pp. 311-319, September 1990.

[Chang 89a] Chang, A.J., DiCesare, F., and Goldbogen, G., "An Algorithm for Constructing a Failure Propagation Tree in Manufacturing

Systems," Proceedings of IEEE Intelligent Controls Conference, Albany, New York, September, 1989, pp. 38-43.

[Chang 89b] Chang, S.J., DiCesare, F., and Goldbogen, G., "The Generation of Diagnostic Heuristics for Automated Error Recovery in Manufacturing Workstations," Proceedings of IEEE Robotics and Automation Conference, Scottsdale, Arizona, May 1989, pp. 522-527.

[de Mello 86] de Mello, L.S.H., and Sanderson, A.C., "AND/OR Graph Representation for Assembly Plans," Proceedings of the AAAI 86 Conference, Philadelphia, Pennsylvania, 1986.

[de Mello 88] de Mello, L.S.H., and Sanderson, A.C., "Planning Repair Sequences Using the AND/OR Graph Representation of Assembly Plans," Proceedings of the IEEE 1988 Applications of Artificial Intelligence, Orlando, Florida, March 1988, pp. 1861-1862.

[DiCesare 88] DiCesare, F. and Jeng, M.D., "A Review of Synthesis Techniques for Petri Nets, "CIM Internal Document, Rensselaer Polytechnic Institute, Troy, New York, 1988.

[Farah 92a] Farah, J.J., and Kelley, R.B., "The Planning Coordinator For Robust Error Recovery And Dynamic On-Line Planning Of Robotic Tasks," Proceedings of the 1992 International Conference On Intelligent Robots and Systems, Raleigh, North Carolina, pp. 1262-1269, July 1992.

[Farah 92b] Farah, J.J., and Kelley, R.B., "Utilizing Semantic Networks to Database and Retrieve Generalized Stochastic Colored Petri Nets," Proceedings of the Fourth Annual Conference on Intelligent Robotic Systems for Space Exploration, Troy, New York, September 1992.

[Farah 92c] Farah, J.J. and Kelley, R.B., "Identifying Relational Error Recovery / On-Line Plans Utilizing Fuzzy Logic Techniques and Semantic Networks," submitted for publication to the 1993 IEEE International Conference on Fuzzy Logic.

[Farah 92d] Farah, J.J, and Kelley, R.B., "Utilizing Semantic Networks and Fuzzy Logic for the Creation of the Planning Coordinator's Primitive Structure Database," submitted for publication to the 1993 IEEE International Conference on Robotics and Automation.

[Fielding 87] Fielding, P. and DiCesare, F., " A Review of, and New Concepts for, Intelligent Automated Error Recovery in Manufacturing Workstations, " Proceedings of the 1987 IEEE International Symposium on Intelligent Control, Philadelphia, Pennsylvania, January 18-20, 1987.

[Fielding 88] Fielding, P., DiCesare, F., and Goldbogen, G." Error Recovery in Automated Manufacturing Through the Augmentation of Programmed Processes," Journal of Robotic Systems 5, 4, August 1988, pp. 337-362.

[Fillmore 86] Fillmore, C.J., " The case for Case," Universals in Linguistic Theory, (E. Bach and R.T. Harms, Editors), Holt, Reinhart and Winston, New York, 1986.

[Gini 83] Gini, G., and Gini, M., "Towards Automatic Error Recovery in Robot Programs," Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, August 8-12, 1983, pp. 821-823.

[Gini 85b] Gini, M., Doshi, R., Garber, S., Gluch, M., Smith, R., and Zaulkernan, I.," Symbolic Reasoning as a Basis for Automatic Error Recovery in Robots," Technical Report TR 85-24, Department of Computer Science, University of Minnesota, Minneapolis, July 1985.

[Gini 85c] Gini, M. and Smith, R., " Reliable Real-Time Robot Operation Employing Intelligent Forward Recovery," Technical Report TR 85-30, Department of Computer Science, University of Minnesota, Minneapolis, September, 1985.

[Hendler 92] Hendler, J.A., " Massively Parallel Marker-Passing In Semantic Networks, " Computers, Mathematics & Applications, 23, 2-5, pp. 277-291, 1992.

[Hörmann 89] Hörmann, A., "A Petri Net Based Control Architecture for a Multi-Robot System," (Paper does not show where it came from. I'll find out though), 1989.

[Koh 88] Koh, I. and DiCesare, F., "Modular Transformation Methods for Generalized Petri Nets and Their Applications to Automated Systems," Internal Document, Rensselaer Polytechnic Institute, Troy, New York, 1988.

[Kosko 86] Kosko, B., "Fuzzy Cognitive Maps," International Journal of Man Machine Studies, 24, pp. 65-75, January 1986.

[Kosko 87] Kosko, B., " Adaptive Inference in Fuzzy Knowledge Networks," Proceedings of the ICNN, 3, pp. 261-268, 1987.

[Kumaradjaja 89] Kumaradjaja, R.," A Causal Reasoning Model For Plan Generation, Execution, Monitoring and Error Recovery in Automated Manufacturing Systems," Ph.D. Thesis, Department of Electrical, Computer and System Engineering, Rensselaer Polytechnic Institute, Troy, New York, 1989.

[Lee 83a] Lee, M.H., Barnes, D.P., and Hardy, N.W.," A Control and Monitoring System for Multiple-Sensor Industrial Robots," Proceedings of the Third International Conference in Robot Vision and Sensory Controls, Boston, Massachusetts, November 1983.

[Lee 84] Lee, M.H., Barnes, D.P., and Hardy, N.W., "Research into Automatic Error Recovery," U.K. Robotics Research, London, UK, 1984.

[Lee 85] Lee, M.H., Barnes, D.P., and Hardy, N.W., "Research into Error Recovery for Sensory Robots," Sensory Review 5, 4, October 1983, pp. 194-197.

[Lee 87] Lee, D.Y., "An Artificial Intelligence Approach for Handling Errors in Automated Manufacturing," M.S. Thesis, Department of Electrical, Computer, and System Engineering, Rensselaer Polytechnic Institute, Troy, New York, May 1987.

[Looney 88] Looney, C.G., "Fuzzy Petri Nets for Rule-Based Decision making," IEEE Transactions on Systems, Man and Cybernetics, 18, 1, pp. 178-183, 1988.

[MacRandal 88] MacRandal, D., "Semantic Networks," Approaches to Knowledge Representation: An Introduction, Research Studies Press / John Wiley & Sons, Letchworth, Hertfordshire / New York, 1988.

[Masterman 62] Masterman, M., "Semantic message detection for machine translation, using an interlingua", 1961 International Conference on Machine Translation of Languages and Applied Language Analysis - Proceedings, Her Majesty's Stationary Office, London, 1962.

[Murata 84] Murata, T., "Modeling and Analysis of Concurrent Systems," Handbook of Systems Engineering, C.R. Vick and C.V.

Ramamoorthy, Editors, New York:Van Nostrand Reinhold, 1984, chapter 3.

[Murata 89] Murata, T., and Komoda, N., " Real Time Control Software for Transaction Processing Based on Colored Safe Petri Net Model," The Journal of Real Time Systems, 1, pp. 299-312, 1989.

[Narahari 88] Narahari, Y., and Viswanadham, N., "Stochastic Petri Net Models for Performance Evaluation of Automated Manufacturing Systems," Elsevier Science Publishers B.V. (North Holland), 1988.

[Nilsson 80] Nilsson, N.J., Principles of Artificial Intelligence, Morgan-Kaufman, California, 1980, chapter 9.

[Norvig 89] Norvig, P., "Marker Passing as a weak method for text inferencing," Cognitive Science, 13, 4, 1989.

[Peterson 81] Peterson, J.L., "Petri Net Theory and the Modeling of Systems," Englewood Cliff: Prentice-Hall, 1981.

[Quillian 68] Quillian, R.M., "Semantic Memory," Semantic Information Processing (M. Minsky, Editor), pp. 216-270, The MIT Press, Cambridge, Massachusetts, 1968.

[Quillian 69] Quillian, M.R., "The teachable language comprehender: A Simulation program and theory of language," Communications of the ACM, 12, pp. 459-476, 1969.

[Richens 56a] Richens, R. H., "Preprogramming for Mechanical-Translation," Mechanical Translation, 3, 1, 1956.

[Richens 56b] Richens, R. H., "Report on research of the Cambridge Language Unit," Mechanical Translation, 3, 2, 1956.

[Riesbeck 86] Riesbeck, C. and Martin, C., "Direct memory accessing parsing," Experience, Memory & Reasoning (J. Kolodner and C.K. Riesbeck, Editors), Lawrence Erlbaum, 1986.

[Ritchie and Thompson 74] Ritchie, D.M. and Thompson, K., "The UNIX Time-Sharing System," Communications of the ACM, 17, 7, pp. 365-375, July 1974.

[Sanderson 87a] Sanderson, A.C. Peshkin, M.A., and de Mello, L.S.H., "Task Planning for Robotic Manipulation in Space Applications," submitted to IEEE Transactions on Aerospace and Electronic Systems, November, 1987.

[Sanderson 87b] Sanderson, A.C., and de Mello, L.S.H., "Task Planing and Control Synthesis for Flexible Assembly Systems," NATO ASI Series: Machine Intelligence and Knowledge Engineering for Robotic Applications, Springer-Verlag, Berlin, 1987.

[Saridis 77] Saridis, G.N., "Self-organizing Controls of Stochastic Systems," Marcel Dekker, New York, 1977.

[Saridis 79] Saridis, G.N., "Toward the Realization of Intelligent Controls," IEEE Proceedings, Vol. 67, No. 8, 1979.

[Saridis 83] Saridis, G.N., "Intelligent Robotic Control," IEEE Transactions on Automatic Control, 29, 4, 1983.

[Saridis 85] Saridis, G.N., "Foundations of The Theory of Intelligent Control," Proceedings IEEE Workshop on Intelligent Controls, pp. 23, Rensselaer Polytechnic Institute, Troy, New York, 1985.

[Saridis 88] Saridis, G.N., "Intelligent Machines : Distributed versus Hierarchical Intelligence," Proceedings IFAC/IMAC International Symposium on Distributed Intelligence Systems (Varna, Belgium), pp. 34-39, 1988.

[Shastri 88] Shastri, L., Semantic Networks: An Evidential Formalization and its Connectionist Realization, Morgan-Kaufman, California, 1988.

[Silberschatz 88] Silberschatz A., and Peterson, J.L., Operating Systems Concepts, Addison-Wesley, New York, 1988, chapters 11 and 12.

[Simmons 73] Simmons, R.F., "Semantic Networks: Their computation and use for understanding English sentences," Computer Models of

Thought and Language, (R. Schank and K.M. Colby, Editors), W.H. Freeman, Chicago, 1973.

[Smith 86] Smith, R. and Gini, M., " Robot tracking and Control Issues in an Intelligent Error Recovery System," Proceedings of the 1986 IEEE International Conference on Robotics and Automation, IEEE Computer Society, 4, 7, pp. 1070-1075, 1986.

[Srinivas 76] Srinivas, S., "Error Recovery in Robots," Ph.D. Thesis, Department of Computer Science, California Institute of Technology, Pasadena, California, 1976.

[Srinivas 78] Srinivas, S., "Error Recovery in Robots Through Failure Reason Analysis, "AFIPS Conference Proceedings, 1978 National Computer Conference, AFIPS Press, Montvale, New Jersey, 1978, pp. 275-282.

[Styblinski 88] Styblinski, M.A., and Meyer, B.D., "Fuzzy Cognitive Maps, Signal Flow Graphs and Qualitative Circuit Analysis," Proceedings of the ICNN, 2 pp. 549-556, 1988.

[Thompson 78] Thompson, K., "UNIX Implementation," The Bell System Technical Journal, 57, 6, part 2, pp. 1931-1946, July/August 1978.

[Thompson 92] Thompson, K., Electronic Mail Correspondence, October 1992.

[Watson 89] Watson, J.F. III, "A Comparison of Performance Evaluation Methodologies for Manufacturing Systems," M.S. Thesis, Department of Electrical Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, New York, 1989.

[Watson 91a] Watson, J.F. III, and Desrochers, A.A., "Applying Generalized Stochastic Petri Nets to Manufacturing Systems Containing Non-Exponential Transition Functions," IEEE Transactions on Systems, Man and Cybernetics, 6:1008-1017, September/October, 1991.

[Watson 91b] Watson, J.F. III, and Desrochers, A.A., "Applying Generalized Stochastic Petri Nets to Manufacturing Systems Containing Non-Exponential Transition Functions, "Proceedings of 1991 IEEE Robotics and Automation Conference, pp. 366-371, Sacramento, California, April 1991.

[Watson 92a] Watson, J.F. III, and Desrochers, A.A., "Methods for Estimating State-Space Size of Petri Nets," Proceedings of 1992 IEEE Robotics and Automation Conference, Nice, France, May 1992.

[Watson 92b] Watson, J.F. III, and Desrochers, A.A., "State Space Size Estimation of Conservative Petri Nets," Proceedings of 1992 IEEE Robotics and Automation Conference, Nice, France, May 1992.

[Williams 86] Williams, D.J. Rogers, P. and Upton, D.M., "Programming and Recovery on Cells for Factory Automation," The International

Journal of Advanced Manufacturing Technology 1,2, pp. 37-47, 1986.

[Yu 90] Yu, Y., and Simmons, R., "Truly parallel understanding of text," Proceedings of AAAI-90, July 1990.

[Zadeh 84] Zadeh, L.A., "Making computers think like people," IEEE Spectrum, pp. 26 - 32, August 1984.

[Zadeh 84a] Zadeh, L.A., "Crisp Sets and Fuzzy Sets", from Fuzzy Logic Text (Must find exact book name), chapter 1, 1984.

[Zhou 88] Zhou, M.C., and DiCesare, F., "Adaptive Design of Petri Net Controllers for Automatic Error Recovery," Proceedings of the Third IEEE International Conference on Intelligent Control, Arlington, Virginia, August 24-25, 1988.

[Zhou 89] Zhou, M.C., and DiCesare, F.," Adaptive Design of Petri Net Controllers for Error Recovery in Automated Manufacturing Systems," IEEE Transactions on Systems, Man and Cybernetics, 19, 5, 1989.

Bibliography

Akatsu, M. , Murata, T., and Kurihara, K., "Verification of Error Recovery Specification for Distributed Data by Using Colored Petri Nets," IEICE Transactions, E 74, 10, pp. 3159 - 3167, October 1991.

Albus, J.S., "A new approach to manipulation control: the cerebellar model articulation controller," Transactions of the ASME, Journal of Dynamics Systems, Measurement and Control, 97, pp. 220-227.

Albus, J.S., Mc Lean, C.E., Barbera, A.J. and Fitzgerald, M.L., " Hierarchical Control for Robots in an Automated Factory," Proceedings of the Thirteenth ISIR Robots 7, 2, 13:29-43, April 1983.

Albus, J.S., "A Theory of Intelligent Systems," Proceedings of the 1990 IEEE Conference on Intelligent Control, Philadelphia, Pennsylvania, September 1990.

Albus, J.S., "The Role of World Modeling and Value Judgment in Perception," Internal Document, National Institute of Standards and Technology, May 1990.

Bonissone, P. , "Expert Systems in Computer Engineering," Fuzzy Sets and Expert Systems Class Notes, Rensselaer Polytechnic Institute, Department of Electrical Computer and System Engineering, Troy, New York.

Brachman, R.J., "On the epistemological status of semantic networks," Associative Networks, (N.V. Findler, Editor),Academic Press, Florida (1975).

Reprinted (1985, Readings in Knowledge Representations, Morgan-Kaufman, California.

Brooks, R.A., "Symbolic Error Analysis and Robot Planning," International Journal of Robotics Research 1,4, 1982.

Brooks, R.A., "Intelligence Without Reason," Proceedings of the 1990 AIAA Conference, Sydney, Australia, pp. 570-590, 1990.

Brooks, R.A.," A Robust Layered Control System for a Mobile Robot," IEEE Journal of Robotics and Automation 2, pp. 14-23, April 1986.

Cao, T. and Sanderson, A.C., "Task Sequence Planning in a Robot Workcell Using AND/OR Nets," Proceedings of the IEEE International Symposium on Intelligent Control, pp. 239-244, Arlington, Virginia, August 1991.

Cao, T. and Sanderson, A.C.," Sensor-based Error Recovery for Robotic Task Sequences Using Fuzzy Petri Nets," Proceedings of the 1992 IEEE Robotics and Automation Conference, Nice, France, May 1992.

Chen, N.S., Yu, H.P., and Huang, S.T., "A self-stabilizing algorithm for constructing spanning trees," Information Processing Letters, 39, pp. 147-151, 1991.

Chen, S., Ke, J. and Chang, J., "Knowledge Representation Using Fuzzy Petri Nets," IEEE Transactions on Knowledge and Data Engineering 2, 3, pp. 311-319, September 1990.

Chang, A.J., DiCesare, F., and Goldbogen, G., "An Algorithm for Constructing a Failure Propagation Tree in Manufacturing Systems," Proceedings of IEEE Intelligent Controls Conference, Albany, New York, September, 1989, pp. 38-43.

Chang, S.J., DiCesare, F., and Goldbogen, G., "The Generation of Diagnostic Heuristics for Automated Error Recovery in Manufacturing Workstations," Proceedings of IEEE Robotics and Automation Conference, Scottsdale, Arizona, May 1989, pp. 522-527.

de Mello, L.S.H., and Sanderson, A.C., "AND/OR Graph Representation for Assembly Plans," Proceedings of the AAAI 86 Conference, Philadelphia, Pennsylvania, 1986.

de Mello, L.S.H., and Sanderson, A.C., "Planning Repair Sequences Using the AND/OR Graph Representation of Assembly Plans," Proceedings of the IEEE 1988 Applications of Artificial Intelligence, Orlando, Florida, March 1988, pp. 1861-1862.

DiCesare, F. and Jeng, M.D., "A Review of Synthesis Techniques for Petri Nets, "CIM Internal Document, Rensselaer Polytechnic Institute, Troy, New York, 1988.

Donald, B.R., "A Geometric Approach to Error Detection and Recovery for Robot Motion Planning with Uncertainty," Artificial Intelligence, 37, Elsevier Science Publishers B.V., (North Holland), pp. 223-271, 1988.

Evens, M.W., Relational models of the lexicon: Representing Knowledge in Semantic Networks, Cambridge University Press, New York, 1988.

Farah, J.J., and Kelley, R.B., "The Planning Coordinator For Robust Error Recovery And Dynamic On-Line Planning Of Robotic Tasks," Proceedings of the 1992 International Conference On Intelligent Robots and Systems, Raleigh, North Carolina, pp. 1262-1269, July 1992.

Farah, J.J., and Kelley, R.B., "Utilizing Semantic Networks to Database and Retrieve Generalized Stochastic Colored Petri Nets," Proceedings of the Fourth Annual Conference on Intelligent Robotic Systems for Space Exploration, Troy, New York, September 1992.

Farah, J.J. and Kelley, R.B., "Identifying Relational Error Recovery / On-Line Plans Utilizing Fuzzy Logic Techniques and Semantic Networks," submitted for publication to the 1993 IEEE International Conference on Fuzzy Logic.

Farah, J.J, and Kelley, R.B., "Utilizing Semantic Networks and Fuzzy Logic for the Creation of the Planning Coordinator's Primitive Structure Database," submitted for publication to the 1993 IEEE International Conference on Robotics and Automation.

Fielding, P. and DiCesare, F., " A Review of, and New Concepts for, Intelligent Automated Error Recovery in Manufacturing Workstations, " Proceedings of the 1987 IEEE International Symposium on Intelligent Control, Philadelphia, Pennsylvania, January 18-20, 1987.

Fielding, P., DiCesare, F., and Goldbogen, G." Error Recovery in Automated Manufacturing Through the Augmentation of Programmed Processes," Journal of Robotic Systems 5, 4, August 1988, pp. 337-362.

Fillmore, C.J., " The case for Case," Universals in Linguistic Theory, (E. Bach and R.T. Harms, Editors), Holt, Reinhart and Winston, New York, 1986.

Freedman, P., "Modeling the actions of an intervention robot," Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Nice, France, pp. 2697-2701, 1992.

Gini, G., and Gini, M., "Towards Automatic Error Recovery in Robot Programs," Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, August 8-12, 1983, pp. 821-823.

Gini, M., Doshi, R., Garber, S., Gluch, M., Smith, R., and Zaulkernan, I.," Symbolic Reasoning as a Basis for Automatic Error Recovery in Robots," Technical Report TR 85-24, Department of Computer Science, University of Minnesota, Minneapolis, July 1985.

Gini, M. and Smith, R., "Reliable Real-Time Robot Operation Employing Intelligent Forward Recovery," Technical Report TR 85-30, Department of Computer Science, University of Minnesota, Minneapolis, September, 1985.

Guo, D., DiCesare, F., and Zhou, M.C., "Moment Generating Function Approach to Performance Analysis of Extended Stochastic Petri Nets," Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, California, 1991.

Hendler, J.A., " Massively Parallel Marker-Passing In Semantic Networks, " Computers, Mathematics & Applications, 23, 2-5, pp. 277-291, 1992.

Hörmann, A., "A Petri Net Based Control Architecture for a Multi-Robot System," (Paper does not show where it came from. I'll find out though), 1989.

Huber, P., Jensen, K., and Shapiro, R.M., "Hierarchies in Coloured Petri Nets," Advances in Petri Nets (1990). Lecture Notes in Computer Science, 483, Springer, New York, pp. 313-341, 1990.

Kasturia, E., DiCesare, F., Desrochers, A., "Real Time Control of Multilevel Manufacturing Systems Using Colored Petri Nets," Proceedings of the 1988 IEEE International Conference on Robotics and Automation, Philadelphia, Pennsylvania, 1988.

Koh, I. and DiCesare, F., "Modular Transformation Methods for Generalized Petri Nets and Their Applications to Automated Systems," Internal Document, Rensselaer Polytechnic Institute, Troy, New York, 1988.

Kosko, B., "Fuzzy Cognitive Maps," International Journal of Man Machine Studies, 24, pp. 65-75, January 1986.

Kosko, B., " Adaptive Inference in Fuzzy Knowledge Networks," Proceedings of the ICNN, 3, pp. 261-268, 1987.

Kumaradjaja, R.," A Causal Reasoning Model For Plan Generation, Execution, Monitoring and Error Recovery in Automated Manufacturing Systems," Ph.D. Thesis, Department of Electrical, Computer and System Engineering, Rensselaer Polytechnic Institute, Troy, New York, 1989.

Lee, M.H., Barnes, D.P., and Hardy, N.W.," A Control and Monitoring System for Multiple-Sensor Industrial Robots," Proceedings of the Third International Conference in Robot Vision and Sensory Controls, Boston, Massachusetts, November 1983.

Lee, M.H., Barnes, D.P., and Hardy, N.W., "Research into Automatic Error Recovery," U.K. Robotics Research, London, UK, 1984.

Lee, M.H., Barnes, D.P., and Hardy, N.W., "Research into Error Recovery for Sensory Robots," Sensory Review 5, 4, October 1983, pp. 194-197.

Lee, D.Y., "An Artificial Intelligence Approach for Handling Errors in Automated Manufacturing," M.S. Thesis, Department of Electrical, Computer, and System Engineering, Rensselaer Polytechnic Institute, Troy, New York, May 1987.

Lehman, F., "Semantic Networks," Computers Mathematics Applications, 23, 2-5, pp. 1-50, 1992.

Looney, C.G., "Fuzzy Petri Nets for Rule-Based Decision making," IEEE Transactions on Systems, Man and Cybernetics, 18, 1, pp. 178-183, 1988.

MacRandal, D., "Semantic Networks," Approaches to Knowledge Representation: An Introduction, Research Studies Press / John Wiley & Sons, Letchworth, Hertfordshire / New York, 1988.

Masterman, M., "Semantic message detection for machine translation, using an interlingua", 1961 International Conference on Machine Translation of Languages and Applied Language Analysis - Proceedings, Her Majesty's Stationary Office, London, 1962.

Murata, T., "Modeling and Analysis of Concurrent Systems," Handbook of Systems Engineering, C.R. Vick and C.V. Ramamoorthy, Editors, New York:Van Nostrand Reinhold, 1984, chapter 3.

Murata, T., and Komoda, N., " Real Time Control Software for Transaction Processing Based on Colored Safe Petri Net Model," The Journal of Real Time Systems, 1, pp. 299-312, 1989.

Narahari, Y., and Viswanadham, N., "Stochastic Petri Net Models for Performance Evaluation of Automated Manufacturing Systems," Elsevier Science Publishers B.V. (North Holland), 1988.

Narahari, Y., and Viswanadham, N., " A Petri Net Approach To The Modeling And Analysis Of Flexible Manufacturing Systems," Annals of Operations Research, 3, pp. 449-472, 1985.

Nilsson, N.J., Principles of Artificial Intelligence, Morgan-Kaufman, California, 1980, chapter 9.

Norvig, P., "Marker Passing as a weak method for text inferencing," Cognitive Science, 13, 4, 1989.

Penders, J.S.J.H., "Error Recovery In A Robot System," Proceedings of the 1989 Conference on Intelligent Autonomous Systems 2, Amsterdam, The Netherlands, 1989.

Peterson, J.L., "Petri Net Theory and the Modeling of Systems," Englewood Cliff: Prentice-Hall, 1981.

Quillian, R.M., "Semantic Memory," Semantic Information Processing (M. Minsky, Editor), pp. 216-270, The MIT Press, Cambridge, Massachusetts, 1968.

Quillian, M.R., "The teachable language comprehender: A Simulation program and theory of language," Communications of the ACM, 12, pp. 459-476, 1969.

Richens, R. H., "Preprogramming for Mechanical-Translation," Mechanical Translation, 3, 1, 1956.

Richens, R. H., "Report on research of the Cambridge Language Unit," Mechanical Translation, 3, 2, 1956.

Riesbeck, C. and Martin, C., "Direct memory accessing parsing," Experience, Memory & Reasoning (J. Kolodner and C.K. Riesbeck, Editors), Lawrence Erlbaum, 1986.

Ritchie, D.M. and Thompson, K., "The UNIX Time-Sharing System," Communications of the ACM, 17, 7, pp. 365-375, July 1974.

Sanderson, A.C. Peshkin, M.A., and de Mello, L.S.H., "Task Planning for Robotic Manipulation in Space Applications," submitted to IEEE Transactions on Aerospace and Electronic Systems, November, 1987.

Sanderson, A.C., and de Mello, L.S.H., "Task Planing and Control Synthesis for Flexible Assembly Systems," NATO ASI Series: Machine Intelligence and Knowledge Engineering for Robotic Applications, Springer-Verlag, Berlin, 1987.

Saridis, G.N., "Self-organizing Controls of Stochastic Systems," Marcel Dekker, New York, 1977.

Saridis, G.N., "Toward the Realization of Intelligent Controls," IEEE Proceedings, Vol. 67, No. 8, 1979.

Saridis, G.N., "Intelligent Robotic Control," IEEE Transactions on Automatic Control, 29, 4, 1983.

Saridis, G.N., and Graham, J.H., "Linguistic Decision Schemata for Intelligent Robots," Pergamon Press Limited, 20,1 pp. 121 - 126, 1984.

Saridis, G.N., "Foundations of The Theory of Intelligent Control," Proceedings IEEE Workshop on Intelligent Controls, pp. 23, Rensselaer Polytechnic Institute, Troy, New York, 1985.

Saridis, G.N., "Intelligent Machines : Distributed versus Hierarchical Intelligence," Proceedings IFAC/IMAC International Symposium on Distributed Intelligence Systems (Varna, Belgium), pp. 34-39, 1988.

Shastri, L., Semantic Networks: An Evidential Formalization and its Connectionist Realization, Morgan-Kaufman, California, 1988.

Silberschatz A., and Peterson, J.L., Operating Systems Concepts, Addison-Wesley, New York, 1988, chapters 11 and 12.

Simmons, R.F., "Semantic Networks: Their computation and use for understanding English sentences," Computer Models of Thought and Language, (R. Schank and K.M. Colby, Editors), W.H. Freeman, Chicago, 1973.

Smith, R. and Gini, M., " Robot tracking and Control Issues in an Intelligent Error Recovery System," Proceedings of the 1986 IEEE International Conference on Robotics and Automation, IEEE Computer Society, 4, 7, pp. 1070-1075, 1986.

Sowa, J.F., Editor, Principles of Semantic Networks: Explorations in the Representation of Knowledge, Morgan Kaufman, California, 1992.

Srinivas, S., "Error Recovery in Robots," Ph.D. Thesis, Department of Computer Science, California Institute of Technology, Pasadena, California, 1976.

Srinivas, S., "Error Recovery in Robots Through Failure Reason Analysis, "AFIPS Conference Proceedings, 1978 National Computer Conference, AFIPS Press, Montvale, New Jersey, 1978, pp. 275-282.

Styblinski, M.A., and Meyer, B.D., "Fuzzy Cognitive Maps, Signal Flow Graphs and Qualitative Circuit Analysis," Proceedings of the ICNN, 2 pp. 549-556, 1988.

Thompson, K., "UNIX Implementation," The Bell System Technical Journal, 57, 6, part 2, pp. 1931-1946, July/August 1978.

Thompson, K., Electronic Mail Correspondence, October 1992.

Tyrrell, A.M., and Holding, D.J., "Design of Reliable Software in Distributed Systems Using the Conversation Scheme," IEEE Transactions on Software Engineering, SE-12, 9, September 1986.

Vogler, W., " Failure Semantics Based On Interval Semiwords Is A Congruence For Refinement," Distributed Computing, 4, pp. 139-162, 1991.

Watson, J.F. III, "A Comparison of Performance Evaluation Methodologies for Manufacturing Systems," M.S. Thesis, Department of Electrical Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, New York, 1989.

Watson, J.F. III, and Desrochers, A.A., "Applying Generalized Stochastic Petri Nets to Manufacturing Systems Containing Non-Exponential Transition Functions," IEEE Transactions on Systems, Man and Cybernetics, 6:1008-1017, September/October, 1991.

Watson, J.F. III, and Desrochers, A.A., "Applying Generalized Stochastic Petri Nets to Manufacturing Systems Containing Non-Exponential Transition Functions, "Proceedings of 1991 IEEE Robotics and Automation Conference, pp. 366-371, Sacramento, California, April 1991.

Watson, J.F. III, and Desrochers, A.A., "Methods for Estimating State-Space Size of Petri Nets," Proceedings of 1992 IEEE Robotics and Automation Conference, Nice, France, May 1992.

Watson, J.F. III, and Desrochers, A.A., "State Space Size Estimation of Conservative Petri Nets," Proceedings of 1992 IEEE Robotics and Automation Conference, Nice, France, May 1992.

Williams, D.J. Rogers, P. and Upton, D.M., "Programming and Recovery on Cells for Factory Automation," The International Journal of Advanced Manufacturing Technology 1,2, pp. 37-47, 1986.

Yu, Y., and Simmons, R., "Truly parallel understanding of text," Proceedings of AAAI-90, July 1990.

Zadeh, L.A., "Making computers think like people," IEEE Spectrum, pp. 26 - 32, August 1984.

Zadeh, L.A., "Crisp Sets and Fuzzy Sets", from Fuzzy Logic Text (Must find exact book name), chapter 1, 1984.

Zhou, M.C., and DiCesare, F., "Adaptive Design of Petri Net Controllers for Automatic Error Recovery," Proceedings of the Third IEEE International Conference on Intelligent Control, Arlington, Virginia, August 24-25, 1988.

Zhou, M.C., and DiCesare, F.," Adaptive Design of Petri Net Controllers for Error Recovery in Automated Manufacturing Systems," IEEE Transactions on Systems, Man and Cybernetics, 19, 5, 1989.